A BYTE BOOK

A Micro Text/McGraw-Hill Copublication

>$13.95

SCIENCE & ENGINEERING PROGRAMS FOR THE TIMEX/SINCLAIR 1000

CASS LEWART

SPECTRUM COMPATIBLE

BYTE BOOKS LEWART SCIENCE & ENGINEERING PROGRAMS FOR THE TIMEX/SINCLAIR 1000 Micro Text / McGraw-Hill

SCIENCE AND ENGINEERING PROGRAMS FOR THE TIMEX/SINCLAIR 1000

SCIENCE AND ENGINEERING PROGRAMS FOR THE TIMEX/SINCLAIR 1000

Cass Lewart

A Micro Text/McGraw-Hill Copublication
New York, N.Y.

I am dedicating this book to my family who allotted me sufficient time on our home computers to be able to complete this book and who provided me with help and many valuable ideas.

Library of Congress Catalog Card Number: 82-062818

ISBN: 0-07-037444-9

Micro Text Publications, Inc.
McGraw-Hill Book Company
1221 Avenue of the Americas
New York, N.Y. 10020

Contents

Preface

The inexpensive Timex/Sinclair 1000 computer is often purchased as a first entry into the computer field. But its toy-like appearance discourages many serious prospective buyers who do not realize its considerable potential. They try a few demonstration programs, find the computer awkward to use, and put it aside. This book, written specifically for the Timex/Sinclair 1000, opens up new fields of applications for the high school or college student, and the engineer or scientist.

A high school or college student will recognize many problems treated here, such as complex arithmetic, combinatorics, distributions, and solutions to transcendental and differential equations, as the subjects of his or her homework assignments. And, the professional will find that with these programs many mathematical and engineering problems can be easily solved on the little computer.

Science and Engineering Programs for the Timex/Sinclair 1000 contains a collection of progams I developed over a number of years for use in my profession as a communications engineer and for use in my hobbies of home computing and electronic design. They cover assorted problems in the fields of electrical engineering, probability, statistics, queuing theory, reliability, curve fitting, graph generation, number theory, computer science, artificial intelligence, and other disciplines.

Some of these programs were originally developed for the Hewlett-Packard HP-25 and HP-67 programmable calculators and were later translated into BASIC. In the process of translation I added a number of features relating to better prompting and input error trapping. Many more useful programs were written in BASIC originally and added to the collection. A few of the programs have been published in magazines. Where this is the case a proper acknowledgement to the magazine is given.

The coding here has been developed from earlier, less sophisticated versions designed to run on the Radio Shack/Sharp Pocket Computers. Translation into Timex/Sinclair 1000 BASIC required many changes, as the Sharp and Timex/Sinclair BASIC languages are only partially compatible. Many programs were completely rewritten and enhanced to make use of advanced Timex/Sinclair 1000 features such as string manipulation and screen display. In the course of writing the Timex/Sinclair 1000 programs I developed many practical programming hints which are detailed in the Appendix of this book. These hints, and various programming techniques described in the Program Remarks section of each chapter, should further sharpen the reader's programming skills.

All programs in *Science and Engineering Programs for the Timex/Sinclair 1000* are specifically written for the computer with at least 2K of RAM or a similarly configured ZX-81. However, each listing is preceded by a program description and the underlying equations, which should make translation to other BASIC computers quite easy. When translating into other computer languages be aware of the idiosyncrasies of each language. For example, the integer function INT(-3.5) will return -4.0 in BASIC, but it will return -3.0 in Fortran!

I hope you will find these programs as useful as I do, and that they will let you enjoy your Timex/Sinclair 1000 even more.

Cass Lewart, Holmdel, NJ, 1983

Introduction

Each program chapter consists of five sections:

1. Program Description. The Program Description provides the reader with the necessary background and understanding of the problem.
2. Instructions. The Instructions demonstrate the key sequences required to execute the program on the Timex/Sinclair 1000.
3. Examples. Examples further clarify the use of the program. This section includes paragraphs on problem definition, discussion of results, and a sample listing of the program run directly from the computer by means of the COPY command.
4. Programming Remarks. The remarks explain program flow and programming techniques. They also substitute for the REMark statements in the listings omitted because of tight memory limitations on the 2K computer.
5. Program Listings. Printed with the LLIST command, and interfaced to a standard dot matrix printer, each listing corresponds exactly to the program which generated the examples.

You will notice that all programs contained in this book have certain common features. They are easy to use and provide comprehensive prompting and error trapping. The programs are not trivial; e.g., they do not do something which could be done more easily with a hand calculator. With very few exceptions, each program effortlessly performs functions which would be difficult to perform in any other way. None of the programs is simply a formula translation. And, they do not use undocumented tricks to save a byte of memory here and there at the cost of clarity.

Before keying in the programs you may want to read the Appendix section describing various programming hints. And finally, last but not least, read the User's Manual that comes with the computer. You can always find something useful in it.

ELECTRICAL ENGINEERING

1. Resonant LC Circuit Parameters

PROGRAM DESCRIPTION

This program evaluates a resonant circuit having a lossless, single-layer airwound coil and a capacitor. The circuit is fully described by two equations—an approximation formula for the coil inductance and a resonance equation of an LC circuit.

Formula 1-1 $(2\pi F)^2 \times LC = 1$ $$L = \frac{(ND)^2 \times 10^{-6}}{18\,D + 40\,A}$$

F is the frequency in Hertz, C is the capacitance in Farads, L is the coil inductance in Henrys, A is the coil length in inches, D is the coil diameter in inches, and N is the number of turns of the coil.

The program helps in designing resonant circuits, winding coils, and repairing or modifying radio receivers and transceivers. When running the program first input all known parameters and then let the computer determine the missing ones by means of the above formulas. As shown in the examples, the program will solve fairly complicated problems. The parameters F, C, L, A, D, or N may be entered, computed, or reviewed in arbitrary order by pressing the appropriate keys.

INSTRUCTIONS

The program prompts with a question mark (?) for a command or with the name of the function when it is expecting a value. When prompted press one of the following keys:

S	Clear all variables, start a new run
F	Enter frequency F in Hertz
Z F	Compute frequency from C and L
C	Enter capacitance C in Farads
Z C	Compute capacitance from F and L
L	Enter coil inductance L in Henrys
Z L	Compute inductance from F and C
Z X, or X	Compute coil inductance from A, D, and N
A	Enter coil length A in inches
Z A	Compute coil length from L, D, and N
D	Enter coil diameter D in inches
Z D	Compute coil diamter from L, A, and N
N	Enter number of turns N for the coil
Z N	Compute number of turns from L, A, and D

To review value of any parameter, key Z followed by F, C, L, A, D, or N.

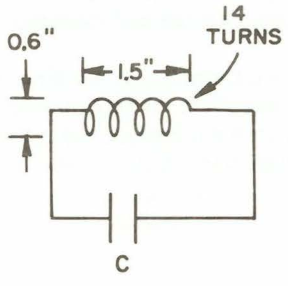
EXAMPLES

Problem Definition

1. Design a resonant circuit using a variable capacitor with capacitance in the range of 35-pF to 350-pF and a coil to be wound on a form 0.6-in. long and 0.3-in. in diameter. The lowest frequency should be approximately 6-MHz. Find the number of turns on the coil and the highest frequency at which the circuit will resonate with the capacitor at its minimum value.

2. A single-layer coil 1.5-in. long, 0.6-in. in diameter, with 14 turns as shown in Figure 1-1 resonates with an unknown capacitor at 10-MHz. What parallel capacitor value is required to lower the resonant frequency to 9-MHz?

Figure 1-1



Sample Run

```
?F
FREQ.?
FREQ.=6000000
?C
CAP.?
CAP.=3.5E-10
?Z
COMPUTE
?L
INDUCT.=2.0103409E-6
?A
LENGTH?
LENGTH=0.6
?D
DIAM.?
DIAM.=0.3
?Z
COMPUTE
?N
TURNS=25.626381
?N
TURNS?
TURNS=25
?X
INDUCT.=1.9132653E-6
?Z
COMPUTE
?F
FREQ.=6150331.3
?C
CAP.?
CAP.=3.5E-11
?Z
COMPUTE
?F
FREQ.=19449055
```

```
?F
FREQ.?
FREQ.=10000000
?A
LENGTH?
LENGTH=1.5
?D
DIAM.?
DIAM.=0.6
?N
TURNS?
TURNS=14
?X
INDUCT.=9.9661017E-7
?Z
COMPUTE
?C
CAP.=2.5416453E-10
?

LET T=C

?F
FREQ.?
FREQ.=9000000
?Z
COMPUTE
?C
CAP.=3.1378337E-10
?

PRINT C-T
5.9618838E-11
```

Discussion of Results

The first problem yields 25.6 turns. In a practical application this number is rounded down to 25 turns. The recomputed resonant frequency is then 6.15-MHz. The highest frequency at which the circuit will resonate with a 35-pF capacitor is then found to be 19.44-MHz.

In the second problem we find a resonant capacitor of 254-pF at 10-MHz. We then stop program execution with BREAK and store this value temporarily as an arbitrary variable T. Program execution is then resumed with the CONT key and the resonating capacitance at 9-MHz is found (313-pF). The program is stopped again and the difference is found to be 59-pF, the final answer. This example illustrates alternate use of the Timex/Sinclair 1000 in the programming and immediate (calculator) mode.

PROGRAMMING REMARKS

The program illustrates the use of the INKEY$ sequence starting in line 40. The CODE function in line 95 returns a number (see page 137-139 of the User's Manual) for each key pressed during the INKEY$ sequence. The program flow is then directed to a line number corresponding to the key pressed (line 105). If the key is not recognized an error trapping routine returns a new prompt in line 30. A flag (Z) is set to 0 or 1 depending on whether the program expects an input (frequency, inductance, etc.) or whether it should find a missing parameter from the given ones.

PROGRAM LISTING

```
 20 CLS
 22 LET Z=0
 25 SLOW
 27 PAUSE 20
 30 PRINT "?";
 40 IF INKEY$="" THEN GOTO 40
 45 PAUSE 10
 50 LET A$=INKEY$
 60 PAUSE 5
 65 PRINT A$
 70 IF A$=INKEY$ THEN GOTO 70
 95 LET K=CODE A$
100 IF K=56 THEN GOTO 20
105 IF K=63 OR K=40 OR K=49 OR K=43 THEN
    GOTO K*10
106 IF K=41 OR K=51 OR K=38 OR K=61 THEN
    GOTO K*10
110 GOTO 30
380 IF Z=1 THEN GOTO 384
381 PRINT "LENGTH?"
382 INPUT A
383 GOTO 386
384 LET Z=0
385 LET A=((D*N*1E-3)**2-18*D*L)/40/L
386 PRINT "LENGTH=";A
```

```
387 GOTO 30
400 IF Z=1 THEN GOTO 404
401 PRINT "CAP.?"
402 INPUT C
403 GOTO 406
404 LET Z=0
405 LET C=1/(2*PI*F)**2/L
406 PRINT "CAP.=";C
407 GOTO 30
410 IF Z=1 THEN GOTO 414
411 PRINT "DIAM.?"
412 INPUT D
413 GOTO 416
414 LET Z=0
415 LET D=(9*L+SQR (81*L*L+40*L*A*N*N/1E-6)
    )/N/N*1E6
416 PRINT "DIAM.=";D
417 GOTO 30
430 IF Z=1 THEN GOTO 434
431 PRINT "FREQ.?"
432 INPUT F
433 GOTO 436
434 LET Z=0
435 LET F=1/2/PI/SQR (L*C)
436 PRINT "FREQ.=";F
437 GOTO 30
490 IF Z=1 THEN GOTO 494
491 PRINT "INDUCT.?"
492 INPUT L
493 GOTO 496
494 LET Z=0
495 LET L=1/(2*PI*F)**2/C
496 PRINT "INDUCT.=";L
497 GOTO 30
510 IF Z=1 THEN GOTO 514
511 PRINT "TURNS?"
512 INPUT N
513 GOTO 517
514 LET Z=0
515 GOSUB 1000
516 LET N=SQR (X*L)/D
517 PRINT "TURNS=";N
518 GOTO 30
```

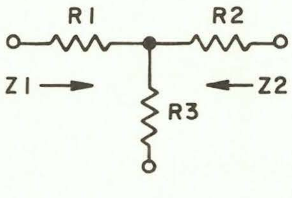
```
 610 LET Z=0
 611 GOSUB 1000
 612 LET L=(N*D)**2/X
 613 GOTO 496
 630 LET Z=1
 631 PRINT "COMPUTE"
 632 GOTO 30
1000 LET X=(18*D+40*A)*1E6
1010 RETURN
```

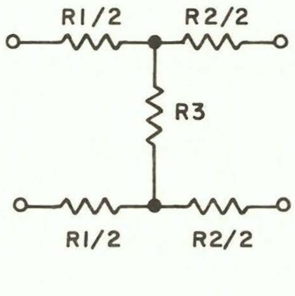
2. Resistive Attenuation and Matching Pads

PROGRAM DESCRIPTION

This program finds resistive component values for attenuation and matching pads commonly used in transmission networks. It computes series and shunt resistors for six types of resistive pads; unbalanced T and Pi, balanced O and H, and the minimum loss matching unbalanced L or balanced C pads. In addition to finding the resistor values, the program also determines the minimum loss of an impedance matching pad. If DC insulation is not required an impedance matching pad is often used as an inexpensive and simple substitute for an impedance matching transformer. Designation of the components is shown in Figures 2-1, 2-2, and 2-3. If the program finds negative resistor values for a requested input impedance, output impedance and loss, then that pad cannot be realized without gain elements and the requested loss should be increased.

Figure 2-1 Unbalanced T/Balanced H





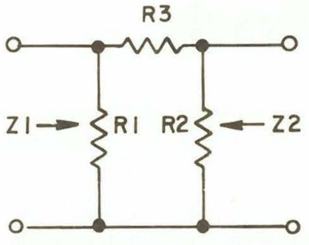
$$N = 10^{\text{LOSS (dB)}/10}$$

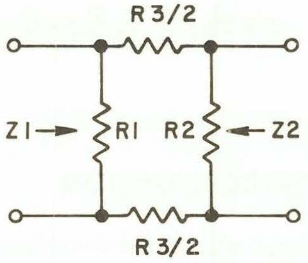
$$R3 = \frac{2\sqrt{N\,Z1 \times Z2}}{N-1}$$

$$R2 = Z2\left(\frac{N+1}{N-1}\right) - R3$$

$$R1 = Z1\left(\frac{N+1}{N-1}\right) - R3$$

Figure 2-2 Unbalanced π, Balanced O



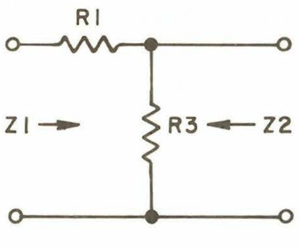


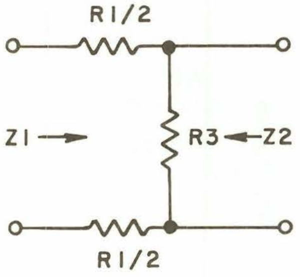
$$R3 = \frac{N - 1}{2} \sqrt{\frac{Z1 \times Z2}{N}}$$

$$\frac{1}{R2} = \frac{1}{Z2}\left(\frac{N + 1}{N - 1}\right) - \frac{1}{R3}$$

$$\frac{1}{R1} = \frac{1}{Z1}\left(\frac{N + 1}{N - 1}\right) - \frac{1}{R3}$$

Figure 2-3 Minimum Loss





$$R1 = Z1 \sqrt{1 - \frac{Z2}{Z1}}$$

$$R3 = \frac{Z2}{\sqrt{1 - \frac{Z2}{Z1}}}$$

$$N = \left(\sqrt{\frac{Z1}{Z2}} + \sqrt{\frac{Z1}{Z2} - 1}\right)^2$$

Loss (db) = 10 log N

$Z1 > Z2$

INSTRUCTIONS

After pressing the RUN key the computer prompts for the type of attentuation/ matching pad. Provide the proper code, 1 for an unbalanced T or a balanced H, 2 for unbalanced Pi or balanced O, or 3 for a minimum loss balanced C or unbalanced L matching pad between different impedances. The next prompts are for Z1 (input impedance), Z2 (output impedance), and loss, except for a minimum loss matching pad, where the loss is computed by the program. Units for all inputs and outputs are ohms and decibels.

EXAMPLES

Problem Definition

1. Given input impedance Z1 = 600-ohm, output impedance Z2 = 900-ohm, find component values for a balanced Pi pad with a loss of 15-db. This type of pad could be encountered in a telephone office.

2. Your TV is overloaded with too strong a signal. Design a balanced O pad (4 resistors) for Z1 = Z2 = 300-ohm and 10-db of loss.

3. Design a minimum loss pad between 75 and 300-ohm impedances. You could use it as a match between a coaxial cable and the terminals of your TV set.

Sample Run

```
ATTENUATION PADS
UNBAL. T, BAL. PI  - ENTER 1
UNBAL. PI, BAL. O  - ENTER 2
MIN. LOSS - ENTER 3?1
ENTER Z1?600
ENTER Z2?900
ENTER LOSS IN DB?15
R1=369.29931 OHM
R2=688.89257 OHM
R3=269.88721 OHM

ATTENUATION PADS
UNBAL. T, BAL. PI  - ENTER 1
UNBAL. PI, BAL. O  - ENTER 2
MIN. LOSS - ENTER 3?2
ENTER Z1?300
ENTER Z2?300
ENTER LOSS IN DB?10
R1=577.48518 OHM
R2=577.48518 OHM
R3=426.90748 OHM
```

```
ATTENUATION PADS
UNBAL. T, BAL. PI - ENTER 1
UNBAL. PI, BAL. O - ENTER 2
MIN. LOSS - ENTER 3?3
Z1 > Z2
ENTER Z1?300
ENTER Z2?75
LOSS=0.57194755 DB
R1=259.80762 OHM
R3=86.60254 OHM

ATTENUATION PADS
UNBAL. T, BAL. PI - ENTER 1
UNBAL. PI, BAL. O - ENTER 2
MIN. LOSS - ENTER 3?
```

Discussion of Results

In general, standard resistor values can be found within a few percent of the computed values. Notice that in the third example the minimum loss is only 0.57-db. This would compare with between 1 and 2-db for a matching transformer.

PROGRAMMING REMARKS

Error checks for the input values are performed in line 75. The proper formulas are then selected depending on the type of pad chosen. To save code the same input sequence (lines 90–125) for Z1 and Z2 is used for all three types of pads. Similarly the same output (print) sequence (lines 350–370) is used for all three branches.

PROGRAM LISTING

```
10 CLS
30 PRINT "ATTENUATION PADS"
40 PRINT "UNBAL. T, BAL. PI - ENTER 1"
50 PRINT "UNBAL. PI, BAL. O - ENTER 2"
60 PRINT "MIN. LOSS - ENTER 3?";
70 INPUT A
72 PRINT A
75 IF A<>INT A OR A>3 OR A<1 THEN GOTO 280
80 IF A=3 THEN PRINT "Z1 > Z2"
90 PRINT "ENTER Z1?";
```

```
100 INPUT B
105 PRINT B
110 PRINT "ENTER Z2?";
120 INPUT C
125 PRINT C
130 IF A=3 THEN GOTO 270
140 PRINT "ENTER LOSS IN DB?";
150 INPUT D
155 PRINT D
160 LET N=10**(D/10)
170 IF A>1 THEN GOTO 220
180 LET G=2*SQR (N*B*C)/(N-1)
190 LET E=B*(N+1)/(N-1)-G
200 LET F=C*(N+1)/(N-1)-G
210 GOTO 350
220 IF A=3 THEN GOTO 270
230 LET G=(N-1)/2*SQR (B*C/N)
240 LET E=1/((N+1)/(N-1)/B-1/G)
250 LET F=1/((N+1)/(N-1)/C-1/G)
260 GOTO 350
270 IF 1-C/B>0 THEN GOTO 300
280 PRINT "ERROR, REENTER"
290 GOTO 40
300 LET E=B*SQR (1-C/B)
310 LET G=C/SQR (1-C/B)
320 LET L=LN (SQR (B/C)+SQR (B/C-1))/LN 10
340 PRINT "LOSS=";L;" DB"
350 PRINT "R1=";E;" OHM"
360 IF A<>3 THEN PRINT "R2=";F;" OHM"
370 PRINT "R3=";G;" OHM"
375 PRINT ""
380 GOTO 20
```

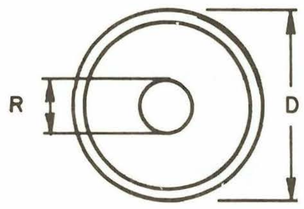
3. Characteristic Impedence of Transmission Lines

PROGRAM DESCRIPTION

This program finds the characteristic impedance for the most common types of transmission lines—the single conductor coaxial, the balanced shielded line, the open two-wire line in air, the parallel strip line used at microwave frequencies, and the transmission line consisting of two shielded parallel wires with sheath return. The five types of transmission lines are shown in the following five figures with the appropriate equations. Units of length are arbitrary (inches, feet, centimeters, etc.), as only ratios of linear measures are used in the equations.

1. Single coaxial line

Figure 3-1

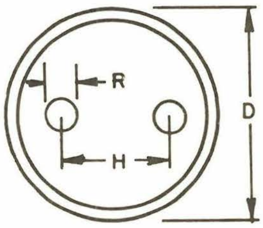


$$Z0 = \frac{138}{\sqrt{\epsilon}} \log \left(\frac{D}{R}\right)$$

ϵ–relative diel. constant

2. Balanced shielded line

Figure 3-2

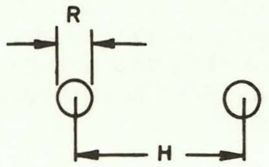


$$Z0 = \frac{276}{\sqrt{\epsilon}} \log \left(2v \frac{1 - \sigma^2}{1 + \sigma^2}\right)$$

$$v = \frac{H}{R} \qquad \sigma = \frac{H}{D}$$

3. Open two-wire line in air

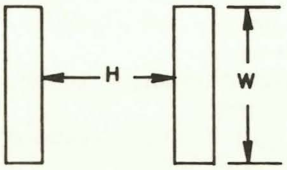
Figure 3-3



$$Z0 = \frac{276}{\sqrt{\epsilon}} \log\left(\frac{H}{R} + \sqrt{\left(\frac{H}{R}\right)^2 - 1}\right)$$

4. Parallel strip line

Figure 3-4



$$Z0 = \frac{377}{\sqrt{\epsilon}} \times \frac{H}{W}$$

$$H/W < 0.1$$

5. Two wires in parallel with sheath return

Figure 3-5 Same Figure as Figure 3-2

$$Z0 = \frac{69}{\sqrt{\epsilon}} \log\left(\frac{v}{2\sigma^2}(1 - \sigma^4)\right)$$

$\sigma = H/D$ $v = H/R$

INSTRUCTIONS

Upon pressing the RUN key the program prompts for the type of transmission line; answer with 1, 2, 3, 4, or 5 referring to Figures 3-1 through 3-5. The next prompt is for the relative dielectric constant of the medium in which the conductor is embedded. This constant is 1.0 for air and is normally supplied by the dielectric manufacturer. Next, the program prompts for the diameter of the inner conductor, spacing between conductors for a two-wire line, and shield diameter when appropriate. Refer to the above figures when entering values. When all inputs are given the program computes the characteristic impedance of the line in ohms.

EXAMPLES

Problem Definition

1. Find characteristic impedance Z0 of a single coaxial line with an inner conductor diameter of 1.5-mm and shield diameter of 6.5-mm. Assume a relative dielectric constant of 1.4 in this and the following examples. This would be a typical coaxial cable used for TV downlead from the antenna.
2. Find Z0 for a balanced shielded cable with conductor spacing of 10-mm, outer diameter of 14-mm and conductor diameter of 1.2-mm.
3. Find Z0 for a two wire line with a spacing of 9-mm and a conductor diameter of 1.5-mm. This would be a typical twin lead cable used for TV.
4. Find Z0 of a parallel strip line, of the type used in microwave transmission, with conductor width of 11-mm and spacing of 1-mm.
5. Finally, find Z0 of two parallel lines with sheath return. The spacing is 10-mm, the outer diameter is 14-mm and the inner diameter is 1-mm.

Sample Run

```
CHAR. IMPEDANCE

ENTER TYPE (1-5)?1
ENTER REL. DIAL. CONSTANT?1.4
SINGLE COAX. LINE
OUTER DIAM D?6.5
COND. DIAM. R?1.5
Z0=74.273381 OHM
```

```
ENTER TYPE (1-5)?2
ENTER REL. DIAL. CONSTANT?1.4
BAL. SHIELDED LINE
SPACING H?10
OUTER DIAM D?14
COND. DIAM. R?1.2
Z0=170.94141 OHM

ENTER TYPE (1-5)?3
ENTER REL. DIAL. CONSTANT?1.4
2 WIRE LINE
SPACING H?9
COND. DIAM. R?1.5
Z0=251.02165 OHM

ENTER TYPE (1-5)?4
ENTER REL. DIAL. CONSTANT?1.4
PARALLEL STRIP LINE
CONDUCTOR WIDTH W?11
SPACING H?1
Z0=28.965741 OHM

ENTER TYPE (1-5)?5
ENTER REL. DIAL. CONSTANT?1.4
2 PAR. LINES, SHEATH RET.
SPACING H?10
OUTER DIAM D?14
COND. DIAM. R?1
Z0=50.1676 OHM
```

Discussion of Results

As shown in the listing of the run the characteristic impedances for the five examples are 74.3, 170.9, 251.0, 29.9, and 50.2-ohm.

PROGRAMMING REMARKS

Note error checks in lines 37, 120, 220, 320, 450, and 520. The same subroutine starting in line 600 is used for input to various branches of the program to save steps.

PROGRAM LISTING

```
 15 CLS
 20 PRINT "CHAR. IMPEDANCE"
 25 PRINT
 27 PRINT "ENTER TYPE (1-5)?";
 30 INPUT A
 35 PRINT A
 37 IF A<>INT A OR A<1 OR A>5 THEN GOTO 60
 40 PRINT "ENTER REL. DIAL. CONSTANT?";
 45 INPUT E
 50 PRINT E
 55 IF E>0 THEN GOTO 100*A
 60 LET L=25
 65 GOTO 720
100 PRINT "SINGLE COAX. LINE"
110 GOSUB 650
120 IF R>0 AND D>R THEN GOTO 150
130 LET L=110
140 GOTO 720
150 LET Z=138/SQR E*LN (D/R)/LN 10
160 GOTO 580
200 PRINT "BAL. SHIELDED LINE"
210 GOSUB 600
220 IF R>0 AND H>R AND D>H+R THEN GOTO 250
230 LET L=210
240 GOTO 720
250 LET V=H/R
260 LET S=H/D
270 LET Z=276/SQR E*LN (2*V*(1-S*S)/
    (1+S*S))/LN 10
280 GOTO 580
300 PRINT "2 WIRE LINE"
310 GOSUB 600
320 IF R>0 AND H>R THEN GOTO 350
330 LET L=310
340 GOTO 720
350 LET S=H/R
360 LET Z=276*LN (S+SQR (S*S-1))/LN 10/SQR E
370 GOTO 580
400 PRINT "PARALLEL STRIP LINE"
410 PRINT "CONDUCTOR WIDTH W?";
420 INPUT W
```

```
430 PRINT W
440 GOSUB 600
450 IF H>0 AND W>0 THEN GOTO 480
460 LET L=410
470 GOTO 720
480 LET Z=377*H/W/SQR E
490 GOTO 580
500 PRINT "2 PAR. LINES, SHEATH RET."
510 GOSUB 600
520 IF R>0 AND H>R AND D>H+R THEN GOTO 550
530 LET L=510
540 GOTO 720
550 LET V=H/R
560 LET S=H/D
570 LET Z=69/SQR E*LN (V/2/S/S*(1-S**4))/LN 10
580 PRINT "ZO=";Z;" OHM"
590 GOTO 25
600 PRINT "SPACING H?";
610 INPUT H
620 PRINT H
630 IF A=3 THEN GOTO 680
640 IF A=4 THEN RETURN
650 PRINT "OUTER DIAM D?";
660 INPUT D
670 PRINT D
680 PRINT "COND. DIAM. R?";
690 INPUT R
700 PRINT R
710 RETURN
720 PRINT "ERROR, REENTER"
730 GOTO L
```

DATA TRANSMISSION

4. Bit Error Rates for Various Modulation Schemes

PROGRAM DESCRIPTION

In a well-designed data transmission link, random noise with Gaussian distribution is the main source of data bit errors. The probability of receiving an incorrect data bit depends on both the Signal-to-Noise (S/N) ratio and the type of data modulation used. In general, circuit complexity is traded for error probability. A more complex circuit with tighter attenuation and delay distortion requirements will result in a lower probability of error for the same S/N ratio. The following program computes the Bit Error Probability (Pe) as a function of S/N for nine popular modulation schemes. The formulas used are as follows:

1. On-Off Keying (OOK), Coherent:

Formula 4-1

$$Pe = \frac{1}{2}\,\mathrm{Erfc}\left(\frac{1}{2}\sqrt{\frac{S}{N}}\right)$$

2. Frequency Shift Keying (FSK), Coherent
   Amplitude Shift Keying (ASK), Coherent
   Pulse Code Modulation (PCM), Unipolar

Formula 4-2

$$Pe = \frac{1}{2}\,\mathrm{Erfc}\sqrt{\frac{1}{2}\,\frac{S}{N}}$$

3. PCM, Polar
   Phase Shift Keying (PSK)

Formula 4-3

$$Pe = \frac{1}{2}\,\mathrm{Erfc}\sqrt{\frac{S}{N}}$$

4. FSK, Noncoherent

Formula 4-4

$$Pe = \frac{1}{2}\,e^{\left(-\frac{S}{2N}\right)}$$

5. Differential Phase Shift Keying (DPSK)

Formula 4-5

$$Pe = \frac{1}{2}\,e^{\left(-\frac{S}{N}\right)}$$

6. ASK, Noncoherent

Formula 4-6

$$Pe = \frac{1}{2}\,\mathrm{Erfc}\left(\frac{1}{2}\sqrt{\frac{S}{N}}\right)$$

The function Erfc (complementary error function) related to the Gaussian noise distribution is defined as follows. The function is computed recursively, as shown in Program 12.

Formula 4-7 $\mathrm{Erf}\ (x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{(-t^2)}\,dt$ $\qquad \mathrm{Erfc}\ (x) = 1 - \mathrm{Erf}\ (x)$

INSTRUCTIONS

When the RUN key is pressed the prograt prompts for the signal-to-noise ratio in db and the type of modulation. Select a S/N ratio and a number between 1 and 6 to indicate the type of modulation as described in the Program Description. The program then computes the bit error probability.

EXAMPLES

Problem Definition

Find the probability of bit error Pe for 12-db S/N for all nine modulation schemes.

Sample Run

```
BIT ERROR PROBABILITY
ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?1
PROB( 12 DB )=.0024385373

ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?2
PROB( 12 DB )=.000034302624

ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?3
PROB( 12 DB )=9.0060101E-9

ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?4
PROB( 12 DB )=.0001808915

ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?5
PROB( 12 DB )=6.544347E-8

ENTER S/N RATIO IN DB?12
SELECT MODULATION CODE 1-6?6
PROB( 12 DB )=.0001990186
```

Discussion of Results

The probability of error for the nine modulation schemes varies widely. The lowest probability of error is given by type 3 (PCM, Polar/PSK) modulation scheme, the highest by type 1 (OOK, Coherent) modulation scheme.

PROGRAMMING REMARKS

The seemingly different formulas are solved by the same section of code through recomputing the argument of the complementary error function. Further discussion of recursive evaluation of the error function is given in Program 12.

PROGRAM LISTING

```
15 CLS
20 PRINT "BIT ERROR PROBABILITY"
```

```
 25 PRINT "ENTER S/N RATIO IN DB?";
 30 INPUT B
 35 PRINT B
 40 LET C=B
 45 PRINT "SELECT MODULATION CODE 1-6?";
 50 INPUT D
 55 PRINT D
 60 IF D>0 AND D<7 AND D=INT D THEN GOTO 100*D
 65 PRINT "ENTRY ERROR"
 70 GOTO 45
100 GOSUB 800
200 GOSUB 800
300 LET X=10**(C/20)
305 LET Y=1/SQR PI*EXP (-X*X)
310 IF X<2 THEN GOTO 350
315 LET A=14/X
320 FOR I=27 TO 1 STEP -1
325 LET A=I/2/(X+A)
330 NEXT I
335 LET Z=Y/(X+A)
340 LET N=1-Z
345 GOTO 700
350 LET I=1
355 LET T=2*X*Y
360 LET N=T
365 LET T=2*X*X*T/(2*I+1)
370 LET M=N+T
375 IF M=N THEN GOTO 393
380 LET N=M
385 LET I=I+1
390 GOTO 365
393 LET Z=1-N
396 GOTO 700
400 GOSUB 800
500 LET Z=EXP -(10**(C/10))
510 GOTO 700
600 LET F=10**(C/10)
610 LET Z=EXP (-F/2)*(1+1/SQR (2*PI*F))
700 LET Z=Z/2
710 PRINT "PROB(";B;" DB)=";Z
720 PRINT
730 GOTO 25
800 LET C=C-10*LN 2/LN 10
810 RETURN
```

NUMBER THEORY

5. Any Base to Any Base Conversion

PROGRAM DESCRIPTION

Converting numbers between different bases is frequently done by computer programmers, mathematicians, high school, and college students. In particular, computer program development requires conversion between binary (2), octal (8), decimal (10), and hexadecimal (16) bases. Number theory problems may require conversion between these and other bases.

Conversion between two arbitrary bases a and b is generally a two-step procedure, converting first from base a to base 10 and then from base 10 to base b. The unique method presented here for positive integers uses the same algorithm for both conversions, thus saving on programming steps.

To represent a number in a base larger than 10, more than one place is necessary to represent one digit in that base. For example 17FE (base 16) would be entered or displayed as [01], [07], [15], [14] or simply 1071514 (1 = 01, 2 = 02,. . . , A = 10, B = 11, C = 12, D = 13, E = 14, F = 15). In general, one place would be reserved per digit for bases 2-10, two places for bases 11-100, etc. The program automatically interprets the entry or display depending on the base. The program has no inherent limitations on the magnitude of the base or the number to be converted. The only limiting factor is the computer's accuracy. For example, the accuracy of the Timex/Sinclair BASIC will limit the input and output to about 10 decimal places. A larger number would result in error.

INSTRUCTIONS

After pressing the RUN key follow the prompts by entering the number to be converted (argument), and the old and the new base. The program will display the argument converted to the new base; if neither the old nor the new base is equal to 10, then the argument in base 10 will also be displayed.

EXAMPLES

1. Convert 17C (base 16) to base 2.
2. Ronvert the decimal equivalent of 17C to base 2 (both results should give the same answer).

3. Convert decimal argument 12345 to base 122.
4. Convert it back from base 122 to base 10.

Sample Run

```
GENERAL BASE CONVERSION
ENTER ARGUMENT?10712
ENTER OLD BASE?16
ENTER NEW BASE?2
10712 TO BASE 16
=101111100 TO BASE 2
=380 TO BASE 10

ENTER ARGUMENT?380
ENTER OLD BASE?10
ENTER NEW BASE?2
380 TO BASE 10
=101111100 TO BASE 2
```

```
ENTER ARGUMENT?12345
ENTER OLD BASE?10
ENTER NEW BASE?122
12345 TO BASE 10
=101023 TO BASE 122

ENTER ARGUMENT?101023
ENTER OLD BASE?122
ENTER NEW BASE?16
101023 TO BASE 122
=3000309 TO BASE 16
=12345 TO BASE 10
```

Discussion of Results

Some interpretation of results is required for bases larger than 10. The result of the third example should be interpreted as [101], [023] (base 122).

PROGRAMMING REMARKS

Extensive error checking of input data is performed in lines 70, 110, and 150. Round-off error is minimized by using the INT function and adding 0.5 to the appropriate functions in lines 440 and 480. Line 480 finds the number of decimal digits required to represent each "digit" to an arbitrary base.

PROGRAM LISTING

```
20 CLS
30 PRINT "GENERAL BASE CONVERSION"
40 PRINT "ENTER ARGUMENT?";
50 INPUT C
60 PRINT C
70 IF C<>INT (ABS C) THEN GOTO 40
80 PRINT "ENTER OLD BASE?";
90 INPUT D
```

```
100 PRINT D
110 IF D<>INT (ABS D) THEN GOTO 80
120 PRINT "ENTER NEW BASE?";
130 INPUT E
140 PRINT E
150 IF E<>INT (ABS E) THEN GOTO 120
160 IF D<>10 THEN GOTO 190
170 LET N=C
180 GOTO 250
190 LET U=D
200 GOSUB 480
210 LET S=C
220 LET Q=D
230 LET R=V
240 GOSUB 390
250 IF E<>10 THEN GOTO 280
260 LET S=N
270 GOTO 340
280 LET U=E
290 GOSUB 480
300 LET S=N
310 LET Q=V
320 LET R=E
330 GOSUB 390
340 PRINT C;" TO BASE ";D
350 PRINT "=";N;" TO BASE ";E
360 IF D<>10 AND E<>10 THEN PRINT "=";S;"
    TO BASE 10"
370 PRINT " "
380 GOTO 40
390 LET M=0
400 LET N=0
410 LET P=S
420 LET T=P
430 LET P=INT (P/R)
440 LET N=N+INT ((T-P*R)*Q**M+.5)
450 IF P=0 THEN RETURN
460 LET M=M+1
470 GOTO 420
480 LET V=INT (10**(1+INT (LN (U-1)/LN 10))+.5)
490 RETURN
```

6. Hex-to-Decimal and Decimal-to-Hex Conversion

PROGRAM DESCRIPTION

This program differs from the previous all-base conversion program in that it converts only between bases 10 and 16. The two conversions (10 to 16 and 16 to 10) are very important for computer program development as most programming tools (assemblers, memory maps, disassemblers, etc.) refer to memory locations and memory contents in either base 10 or 16 (hex). This program, unlike the previous one, uses the standard alphanumeric presentation for hexadecimal digits: 0 through 9 and A, B, C, D, E, and F for digits 10 through 15. The same comments about accuracy and round-off as in the previous program apply.

INSTRUCTIONS

Press the RUN key and follow prompts to perform Hex-to-Decimal (1), or Decimal-to-Hex (2) conversion. After the next prompt enter the argument either in Hex (using A through F if necessary) or as a decimal number.

EXAMPLES

1. Convert F697 (base 16) to base 10.
2. Convert the result back to base 16.
3. Convert FFFF (base 16) to base 10.

Sample Run

```
HEX-TO-DECIMAL, ENTER 1
DECIMAL-TO-HEX, ENTER 2?1
ENTER ARGUMENT IN HEX?F697
=63127 IN BASE 10


HEX-TO-DECIMAL, ENTER 1
DECIMAL-TO-HEX, ENTER 2?2
ENTER ARGUMENT TO BASE 10?63127
=F697 IN BASE 16
```

```
HEX-TO-DECIMAL, ENTER 1
DECIMAL-TO-HEX, ENTER 2?1
ENTER ARGUMENT IN HEX?FFFF
=65535 IN BASE 10
```

Discussion of Results

Prompting and results are self evident.

PROGRAMMING REMARKS

Error checking of input data is performed in lines 70 and 270. Base 16 digits are entered as a character string (A$) and each character in the string is interpreted in line 260. If a character is found which does not correspond to a number between 0 and 15 it is rejected as an error. Translation from decimal numbers into a hex character string occurs in line 510. This program demonstrates the power of BASIC character string manipulation functions such as CHR$, CODE, and LEN.

PROGRAM LISTING

```
 10 DIM B$(10)
 20 CLS
 30 PRINT
 35 PRINT "HEX-TO-DECIMAL, ENTER 1"
 40 PRINT "DECIMAL-TO-HEX, ENTER 2?";
 50 INPUT B
 60 PRINT B
 70 IF B<>1 AND B<>2 THEN GOTO 30
 80 GOTO 200*B
200 PRINT "ENTER ARGUMENT IN HEX?";
210 INPUT A$
220 PRINT A$
230 LET L=LEN A$
240 LET N=0
250 FOR I=0 TO L-1
260 LET C=CODE A$(L-I)-28
270 IF C>=0 AND C<=16 THEN GOTO 310
280 PRINT "BAD ENTRY"
290 PAUSE 200
300 RUN
310 LET N=N+C*16**I
```

```
320 NEXT I
330 PRINT "=";N;" IN BASE 10"
340 GOTO 640
400 PRINT "ENTER ARGUMENT TO BASE 10?";
410 INPUT D
420 PRINT D
430 IF D=INT (ABS D) THEN GOTO 470
440 PRINT "BAD ENTRY"
450 PAUSE 200
460 RUN
470 LET K=0
480 LET K=K+1
485 LET F=D
490 LET D=INT (D/16)
500 LET E=F-16*D
510 LET B$(K)=CHR$ (E+28)
580 IF F>0 THEN GOTO 480
590 PRINT "=";
600 FOR I=1 TO K-1
610 PRINT B$(K-I);
620 NEXT I
630 PRINT " IN BASE 16"
640 PRINT " "
650 GOTO 30
```

7. Prime Factor Decomposition

PROGRAM DESCRIPTION

One of the first problems facing a student in number theory is the prime factor decomposition of an integer. Prime number decomposition is not merely a theoretical exercise—it is applicable in many areas, including cryptography. Some of the modern cryptographic codes are based on the assumption that it is extremely time consuming, even for large computers, to find prime factors of large numbers. In the so-called "public key" technique the encryption code (product of two primes) can be publicized, while the prime factors being the decryption code, are known only to authorized recipients. If a code is based on a key that is a product of two large primes, which it would take years for a large computer to find, then the code is for all practical purposes unbreakable.

The two principal methods for finding prime factors of an integer are the Eratosthenes sieve, named after a famous Greek mathematician, and a simple odd divisor check. The first method is faster but also more appropriate for large computers, as it stores in memory all successive primes smaller than the square root of the number to be decomposed. The second method is much simpler, as it only checks for decomposition into 2, 3, 5, and the successive odd integers. Though the second approach is somewhat wasteful on execution time, since it checks against all odd but not necessarily prime integers, the second method imposes low memory requirements on the computer, because no intermediate results have to be stored. The second method is therefore the only one suitable for calculating primes on the Timex/Sinclair 1000 computer, and is the basis for the following program.

INSTRUCTIONS

Press the RUN key to start the program, then follow the prompt by entering the number to be decomposed into prime factors. The program then displays how many prime factors were found and the prime factors, one by one.

EXAMPLES

Problem Definition

Decompose 12345, 66759, and 71 into prime factors.

Sample Run

```
PRIME FACTOR FINDER
ENTER NUMBER?12345
 NO. OF FACTORS=3
FACTOR NO. 1=3
FACTOR NO. 2=5
FACTOR NO. 3=823
```

```
PRIME FACTOR FINDER
ENTER NUMBER?66759
 NO. OF FACTORS=5
FACTOR NO. 1=3
FACTOR NO. 2=7
FACTOR NO. 3=11
FACTOR NO. 4=17
FACTOR NO. 5=17
```

```
PRIME FACTOR FINDER
ENTER NUMBER?71
 NO. OF FACTORS=1
FACTOR NO. 1=71
```

Discussion of Results

Notice in the second example that 17 appears twice, indicating that it is a multiple factor. In the third example we test a prime number (71) which has a single non-trivial factor, itself.

PROGRAMMING REMARKS

The program performs error checking in line 90. If the number to be decomposed is larger than 1,000,000,000 then round-off errors will not allow the program to operate correctly. For user recall the first 50 prime factors are stored in a dimensioned array B(50) and could be recalled by a user written program. If there are more than 50 factors then the program displays only the first 50.

PROGRAM LISTING

```
20 CLS
25 LET K=0
27 LET D=0
30 DIM B( 50 )
40 PRINT "PRIME FACTOR FINDER"
50 PRINT "ENTER NUMBER?";
60 INPUT A
70 PRINT A
80 LET F=A
```

```
 90 IF A=INT A AND A>1 AND A<1E9 THEN. GOTO 120
100 PRINT "ERROR,REENTER"
110 GOTO 50
120 LET E=A/2
130 LET C=2
140 IF E<>INT E OR 2*E<A THEN GOTO 170
150 GOSUB 320
160 GOTO 120
170 LET D=D+1
180 LET C=1+2*D
190 IF C>SQR A THEN GOTO 250
200 LET E=A/C
210 IF E<>INT E OR E*C<>A THEN GOTO 170
220 LET D=D-1
230 GOSUB 320
240 GOTO 170
250 IF A<=1 THEN GOTO 280
260 LET K=K+1
270 LET B(K)=A
280 PRINT " NO. OF FACTORS=";K
290 FOR I=1 TO K
300 PRINT "FACTOR NO. ";I;"=";B(I)
310 NEXT I
312 PRINT " "
315 GOTO 25
320 IF K>=50 THEN RETURN
330 LET K=K+1
340 LET B(K)=C
350 LET A=E
360 RETURN
```

8. Relative Offset Computations for Machine Code Programming

PROGRAM DESCRIPTION

The purpose of this program is to compute relative addresses of machine code instructions. When writing short programs in machine code for any of the popular 8-bit microprocessors such as Z80, 6502, or 6800 one either has to load the Assembler program, which may not be available, or "hand" assemble the machine code program. The main problem then is to find the arguments of the relative branching instructions. The program shown here will find in hexadecimal notation the argument of a relative branching instruction zz given the absolute address of that instruction, the "FROM" address (X) and the hexadecimal address of the instruction to which the program should branch if the branching condition is true, the "TO" address (Y). For example, given the following code in Z80 machine language:

Address	Instruction	Label	Mnemonic
20FC	DD 7E 00	LOOP	LD A,(IX)
....	....	....	....
2109	B8		CP B
210A	20 zz		JR NZ LOOP

one has to find the value of zz, also known as offset, given the address of the "FROM" branching instruction 210A (hex) and the address of the "TO" instruction 20FC (hex). To compute zz one of the following equations must be solved:

zz = Y − X − 2 for Y > X (forward branch), or
zz = FF (hex) + Y − X − 1 for Y < = X (reverse branch).

The resulting offset in the above example would be F0 (hex) and the last instruction would read 20 F0. Because the offset zz can only assume values between 0 and FF (hex) for 8-bit microprocessors, the relative branching is limited to one-half page (1 page = 256 locations) in the forward or reverse direction. The program checks whether the branching instruction can be performed, as it analyzes the addresses of the FROM/TO instructions. The program then indicates if the relative branching can or cannot be performed in a single step, and if it will result in a forward or a reverse jump.

INSTRUCTIONS

When the RUN key is pressed the program prompts for the address of the "FROM" and the "TO" instructions. Both should be entered in hexadecimal notation.

EXAMPLES

Problem Definition

Example #	Branch Instruction Absolute address (hex)	Destination Instruction Absolute address (hex)
1	123A	127B
2	25F3	2613
3	010A	00FC
4	AA25	BA24

Sample Run

```
REL. ADDRESSING (8-BIT)

ENTER "FROM" ADDRESS IN HEX?123A
ENTER "TO" ADDRESS IN HEX?127B
FORWARD OFFSET=3F

ENTER "FROM" ADDRESS IN HEX?25F3
ENTER "TO" ADDRESS IN HEX?2613
FORWARD OFFSET=1E
```

```
ENTER "FROM" ADDRESS IN HEX?10A
ENTER "TO" ADDRESS IN HEX?FC
REVERSE OR IN PLACE OFFSET=F0

ENTER "FROM" ADDRESS IN HEX?AA25
ENTER "TO" ADDRESS IN HEX?BA24
4095(DEC) OFFSET TOO LARGE,REENT
ER

ENTER "FROM" ADDRESS IN HEX?AA25
ENTER "TO" ADDRESS IN HEX?AA23
REVERSE OR IN PLACE OFFSET=FC
```

Discussion of Results

The first three examples result in hexadecimal offsets of 3F, 1E and F0. The fourth example indicates that the offset is too large. We realize then that the "TO" address should be AA23 instead of BA24 and end up with an offset of FC.

PROGRAMMING REMARKS

Notice the use of double quotes (Shift Q) in lines 45 and 120. To perform arithmetic on hexadecimal numbers the numbers are first converted to base 10 and the results are converted back to base 16. Conversion from decimal to hex and vice versa is performed in the same way as in program #6 by means of string manipulation functions.

PROGRAM LISTING

```
 20 CLS
 30 PRINT "REL. ADDRESSING (8-BIT)"
 40 PRINT
 45 PRINT "ENTER ""FROM"" ADDRESS IN HEX?";
 50 INPUT C$
 60 PRINT C$
 70 GOSUB 330
 80 IF F=0 THEN GOTO 110
 90 PRINT "ERROR, REENTER"
100 GOTO 40
```

```
110 LET J=X
120 PRINT "ENTER ""TO"" ADDRESS IN HEX?";
130 INPUT C$
140 PRINT C$
150 GOSUB 330
160 IF F=0 THEN GOTO 190
170 PRINT "ERROR, REENTER"
180 GOTO 120
190 LET D=X-J
200 IF D<=129 AND D>=-126 THEN GOTO 230
210 PRINT D;"(DEC) OFFSET TOO LARGE,REENTER"
220 GOTO 40
230 IF D<2 THEN GOTO 260
240 LET Z=D-2
245 LET H$="FORWARD"
250 GOTO 270
260 LET Z=254+D
265 LET H$="REVERSE OR IN PLACE"
270 LET G=INT (Z/16)
280 LET E$=CHR$ (G+28)
290 LET W$=CHR$ (Z-16*G+28)
300 PRINT H$;" OFFSET=";E$;W$
320 GOTO 40
330 LET F=0
340 LET X=0
350 LET L=LEN C$
360 FOR I=0 TO L-1
370 LET R=CODE C$(L-I)-28
380 IF R<0 OR R>15 THEN LET F=1
390 LET X=X+R*16**I
400 NEXT I
410 RETURN
```

COMPUTER GENERATED PLOTTING

9. Curve Smoothing and Interpolation / Extrapolation with Lagrange Polynomials

PROGRAM DESCRIPTION

This program is the mathematical equivalent of a "French Curve." Its main application is interpolation and extrapolation of measured results. Given X(i), Y(i), i = 1, . . . ,n, the program finds a unique polynomial of degree (n − 1) fitting these n points, and evaluates the polynomial for arbitrary values of the independent variable X. The approach differs from a least squares approximation in that the resulting polynomial fits the points X(i), Y(i) exactly.

The following formula, attributed to the French mathematician Lagrange, describes the polynomial:

Formula 9-1

$$y(x) = \sum_{i=1}^{n} y_i \times \frac{\prod_{s=1, s \neq i}^{n} (x - x_s)}{\prod_{s=1, s \neq i}^{n} (x_i - x_s)}$$

For practical applications it is advisable to use care in selecting only adjacent points to compute the polynomial so that wildly oscillatory curves are not obtained as extrapolations and interpolations of relatively smooth functions. In general, a cubic polynomial based on four nearby points (n = 4) will be satisfactory.

An important feature of the program is that a moving interpolation "window" can be established so that only k measured values (k < = n) closest to the interpolation point will be included in the computation.

INSTRUCTIONS

Pressing the RUN key initializes the program and issues prompts for the number of points (n < = 20) and the successive X(i) and Y(i) values of points in increasing order of X; the number of points in the "window" and the value of X at which the function should be interpolated or extrapolated. The next option is to recompute the function for another X, change the "window" width, or start a new run.

EXAMPLE

Problem Definition

As shown in Figure 9-1 six points have been measured. Assume a "window" width of four and evaluate the function at X = 2.1 (interpolation) and at X = 7.0 (extrapolation). Then change the "window" width to five points and recompute the same two points.

Sample Run

```
LAGRANGE INTERPOLATION
NO. POINTS?6
ENTER X( 1 )?-0.5
ENTER Y( 1 )?0
ENTER X( 2 )?0.5
ENTER Y( 2 )?0.7
ENTER X( 3 )?2
ENTER Y( 3 )?2
ENTER X( 4 )?3
ENTER Y( 4 )?2.6
ENTER X( 5 )?4.5
ENTER Y( 5 )?2.8
ENTER X( 6 )?6
ENTER Y( 6 )?3.3
NO. POINTS IN WINDOW?4
FIND Y(X), X=?2.3
Y( 2.3 )=2.20996
```

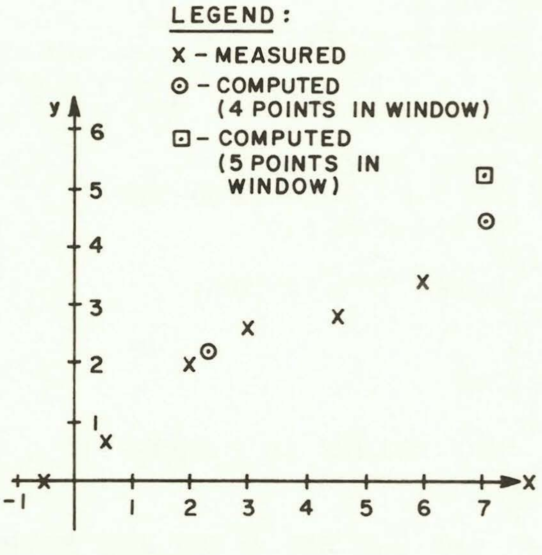
```
NEW X 1,NEW WINDOW 2,NEW RUN 3?1
FIND Y(X), X=?7
Y(7)=4.4333333

NEW X 1,NEW WINDOW 2,NEW RUN 3?2
NO. POINTS IN WINDOW?5
FIND Y(X), X=?2.3
Y(2.3)=2.22256

NEW X 1,NEW WINDOW 2,NEW RUN 3?1
FIND Y(X), X=?7
Y(7)=5.1909091

NEW X 1,NEW WINDOW 2,NEW RUN 3?1
FIND Y(X), X=?4.5
Y(4.5)=2.8
```

Figure 9-1



Discussion of Results

The example shows that "window" width strongly affects extrapolated values. The interpolated value changed only from 2.20. . to 2.22. . , whereas the extrapolated value changed from 4.43. . to 5.19. . . The user has to decide which "window" width is more appropriate. The example also shows that the Lagrange polynomial goes exactly through each given point. When Y(4.5) is computed by the program, it generates 2.8, which is the exact value originally entered for Y(4.5).

PROGRAMMING REMARKS

The program checks for the correct entries in lines 80, 170, and 280. Branching is performed in three simple statements in lines 560-580. The Lagrange formula evaluation takes place in a double loop in lines 430-500.

PROGRAM LISTING

```
  5 CLS
 20 DIM X( 20 )
 30 DIM Y( 20 )
 40 PRINT "LAGRANGE INTERPOLATION"
 50 PRINT "NO. POINTS?";
 60 INPUT A
 70 PRINT A
 80 IF A<=20 AND A>2 AND A=INT ABS A THEN
    GOTO 110
 90 PRINT "ERROR"
100 GOTO 50
110 FOR I=1 TO A
120 PRINT "ENTER X(";I;")?";
130 INPUT X
140 PRINT X
150 LET X( I )=X
160 IF I=1 THEN GOTO 200
170 IF X( I )>X( I-1 ) THEN GOTO 200
180 PRINT "X( I+1 )<=X( I )"
190 GOTO 120
200 PRINT "ENTER Y(";I;")?";
210 INPUT Y
220 PRINT Y
230 LET Y( I )=Y
240 NEXT I
250 PRINT "NO. POINTS IN WINDOW?";
260 INPUT B
270 PRINT B
280 IF B<=A AND B>1 AND B=INT ABS B THEN
    GOTO 310
290 PRINT "ERROR"
300 GOTO 250
310 LET C=INT (B/2)
320 LET I=1
330 PRINT "FIND Y(X), X=?";
340 INPUT W
```

```
350 PRINT W
360 IF W<=X( I ) OR I=A THEN GOTO 390
370 LET I=I+1
380 GOTO 360
390 LET Z=0
400 LET D=I-C
410 IF D<1 THEN LET D=1
420 IF I-C+B>A THEN LET D=A-B+1
430 FOR I=D TO D+B-1
440 LET E=1
450 FOR J=D TO D+B-1
460 IF J=I THEN GOTO 480
470 LET E=E*( W-X( J ))/( X( I )-X( J ))
480 NEXT J
490 LET Z=Z+E*Y( I )
500 NEXT I
510 PRINT "Y( ";W;" )=";Z
520 PRINT
530 PRINT "NEW X 1,NEW WINDOW 2,NEW RUN 3?";
540 INPUT F
550 PRINT F
560 IF F=1 THEN GOTO 310
570 IF F=2 THEN GOTO 250
580 IF F=3 THEN GOTO 5
590 GOTO 530
```

10. Readable Scales for Computer Plots

PROGRAM DESCRIPTION

Every good plotting program should have the ability to automatically calculate scales for the X, Y, or Z axes. If the data to be plotted is generated by the computer, then the user cannot always anticipate the scales needed. The Timex/Sinclair 1000 can, however, automate the task and, moreover, do it intelligently, to come up with "readable" scales. The program, given the range of a variable, produces a readable linear scale with uniform intervals. Output of the program can then be used to drive any plotting routine. A readable linear scale is defined here as a scale with interval width the product of an integer power of 10 and 1, 2, or 5, and scale values that are integer multiples of the interval width.

The above definition permits scale values such as:

−0.5, 0.0, 0.5, 1.0,. . .
1.24, 1.26, 1.28,. . .
100, 200, 300,. . ., etc.,

but would prohibit the following examples:

−1, 4, 9,. . .
1.2, 1.31, 1.42,. . .
0, 4, 8,. . ., etc.

Given a minimum MIN, a maximum MAX of the array to be plotted, and an approximate number of desired scale intervals N, the program computes four parameters, a new minimum MINP, a new maximum MAXP, an interval width DIST, and a number of scale intervals NP which will best fit the plot and approximate N. The computed parameters, besides producing a readable scale, will satisfy the following inequalities:

$(MIN - DIST) < MINP <= MIN,$
$MAX <= MAXP < (MAX + DIST)$ and
$N/\sqrt{2.5} < NP < (N\sqrt{2.5} + 2).$

A unique feature of the program is that it introduces a narrow gate $\pm$ J (line 200) around the MIN and MAX values to avoid an unnecessarily large range between MINP and MAXP due to computer roundoff.

INSTRUCTIONS

Initialize the program by pressing the RUN key, then follow prompts for the minimum and the maximum of your array and for the approximate desired number of scale intervals. The program then returns a new minimum, a new maximum, the interval size, and the number of intervals.

EXAMPLES

Problem Definition

Find the scale values for an array with MIN, MAX, and N, respectively equal to:

1. −3.1, 11.1, 5
2. 345, 510, 20
3. −1100, −520, 50
4. 17.5, 17.9, 10

Sample Run

```
SCALING PROGRAM

ENTER MINIMUM?-3.1
ENTER MAXIMUM?11.1
ENTER APPR. NO. OF INTERVALS?5
NEW MIN=        -4
NEW MAX=        12
NEW INTERVAL=   2
NO. INTERVALS=  8

ENTER MINIMUM?345
ENTER MAXIMUM?510
ENTER APPR. NO. OF INTERVALS?20
NEW MIN=        340
NEW MAX=        510
NEW INTERVAL=   10
NO. INTERVALS=  17

ENTER MINIMUM?-1100
ENTER MAXIMUM?-520
ENTER APPR. NO. OF INTERVALS?50

NEW MIN=        -1100
NEW MAX=        -520
NEW INTERVAL=    10
NO. INTERVALS=   58

ENTER MINIMUM?17.9
ENTER MAXIMUM?17.5
ERROR, MIN>=MAX

ENTER MINIMUM?17.5
ENTER MAXIMUM?17.9
ENTER APPR. NO. OF INTERVALS?10
NEW MIN=         17.5
NEW MAX=         17.9
NEW INTERVAL=    .05
NO. INTERVALS=   8
```

Discussion of Results

The four examples show that the program produces "readable" scales for a wide range of inputs. The first example would result in scale values of −4, −2, 0, 2, . . ., 12. The second example would yield 340, 350, . . ., 510. The third example would result in −1100, −1090, . . ., −520. The fourth example would result in 17.5, 17.55, . . ., 17.9. The fourth example also shows the error message when the minimum and maximum were entered by mistake in reverse order.

PROGRAMMING REMARKS

Error checking occurs in line 100. Determination of scales is done in lines 240-270. The PRINT statement in line 340 also performs computation to save code.

PROGRAM LISTING

```
 20 CLS
 30 PRINT "SCALING PROGRAM"
 40 PRINT
 45 PRINT "ENTER MINIMUM?";
 50 INPUT X
 60 PRINT X
 70 PRINT "ENTER MAXIMUM?";
 80 INPUT Y
 90 PRINT Y
100 IF Y>X THEN GOTO 130
110 PRINT "ERROR, MIN>=MAX"
120 GOTO 40
```

```
130 PRINT "ENTER APPR. NO. OF INTERVALS?";
140 INPUT N
150 PRINT N
160 IF N>0 AND N=INT ABS N THEN GOTO 190
170 PRINT "ERROR"
180 GOTO 130
190 LET D=(Y-X)/N
200 LET J=D/1E5
210 LET E=INT (LN D/LN 10)
220 LET F=D/10**E
230 LET V=10
240 IF F<SQR 50 THEN LET V=5
250 IF F<SQR 10 THEN LET V=2
260 IF F<SQR 2 THEN LET V=1
270 LET C=V*10**E
280 LET G=INT (X/C)
290 IF ABS (G+1-X/C)<J THEN LET G=G+1
300 LET A=C*G
310 LET H=INT (Y/C)+1
320 IF ABS (Y/C+1-H)<J THEN LET H=H-1
330 LET B=C*H
340 PRINT "NEW MIN=",A,"NEW MAX=",B,
    "NEW INTERVAL=",
345 PRINT C,"NO. INTERVALS=",H-G
360 GOTO 40
```

Originally published as "Pocket Computer Scales Computer Generated Plots." Reprinted from *Electronics,* December 15, 1981, copyright © McGraw-Hill, Inc. 1981. All rights reserved.

PROBABILITY AND STATISTICS

11. Combinations, Permutations, and Factorials

PROGRAM DESCRIPTION

This program evaluates the three most important functions in statistics and combinatorics.

Combinations. The program computes the number of possible combinations of N different objects taken from a set of M objects. Changing the sequence of the objects in a combination does not result in a new combination.

The resulting function C(M,N) is also called the binomial coefficient and can be represented as follows:

Formula 11-1

$$C\ (M,N) = \binom{M}{N} = \frac{M!}{(M-N)!\ N!} =$$

$$= \left(\frac{M+1}{N} - 1\right)\left(\frac{M+1}{N-1} - 1\right)\ldots\left(\frac{M+1}{1} - 1\right)$$

The first expression, which is the definition of the binomial coefficient C(M,N), may lead to large errors by dividing a large number by another large number. If any of the factorials in the nominator or denominator exceeds 33! then the first expression could not be evaluated by the Timex/Sinclair 1000 at all. On the other hand the second expression avoids the overflow problem, hence is the one used here to evaluate C(M,N).

Permutations. The program computes the number of permutations possible with M objects taken N at a time. Changing the order of the objects in a permutation results in a different permutation.

The resulting function P(M,N) is computed as follows:

Formula 11-2

$$P(M,N) = M(M - 1)\ldots(M - N + 1)$$

Factorials: A factorial F(M) equals to the number of ways in which M objects can be rearranged. The function is defined and calculated as follows:

Formula 11-3

$$F(M) = M(M - 1)\ldots 1$$

EXAMPLES

Problem Definition

1. In how many ways can 10 people be arranged in a row? Evaluate F(10).
2. How many groups of 3 people can be selected out of a group of 20 when the order in each group of 3 matters? Evaluate P(20,3).
3. How many groups of 3 can be selected out of 20 people, when the order does not matter? Evaluate C(20,3).
4. Same as example 3 but select a group of 17 people out of 20. Evaluate C(20,17).

Sample Run

```
FACTORIALS 1,PERMUTATIONS 2
COMBINATIONS (BIN. COEFF.) 3?1
FACTORIAL OF?10
FACT( 10 )=3628800

FACTORIALS 1,PERMUTATIONS 2
COMBINATIONS (BIN. COEFF.) 3?2
M=?20
N=?3
PERM( 20,3 )=6840
```

```
FACTORIALS 1,PERMUTATIONS 2
COMBINATIONS (BIN. COEFF.) 3?3
M=?20
N=?3
COMB( 20,3 )=1140

FACTORIALS 1,PERMUTATIONS 2
COMBINATIONS (BIN. COEFF.) 3?3
M=?20
N=?17
COMB( 20,17 )=1140

FACTORIALS 1,PERMUTATIONS 2
COMBINATIONS (BIN. COEFF.) 3?
```

Discussion of Results

Example one produces a very large number, examples three and four produce the same result. The reason for the equality is that when one chooses a group of 17, then the remaining group is three— which is the same as example number three.

PROGRAMMING REMARKS

Check of input errors is performed in lines 60 and 130. The same PRINT statement in line 450 is used for both permutations and combinations by assigning a proper name to variable C$. The program makes use of the equality C(M,N) = C(M,M – N) and selects in line 250 the fastest computation path.

PROGRAM LISTING

```
 20 CLS
 30 PRINT
 35 PRINT "FACTORIALS 1,PERMUTATIONS 2"
 37 PRINT "COMBINATIONS (BIN. COEFF.) 3?";
 40 INPUT F
 50 PRINT F
 60 IF F<>1 AND F<>2 AND F<>3 THEN GOTO 30
 70 GOTO 100*F
100 PRINT "FACTORIAL OF?";
110 INPUT M
120 PRINT M
```

```
130 IF M<=0 OR M<>INT ABS M OR M>33
    THEN GOTO 100
140 LET B=1
150 FOR I=1 TO M
160 LET B=B*I
170 NEXT I
180 PRINT "FACT(";M;")=";B
190 GOTO 30
200 LET C$="PERM("
210 GOSUB 480
220 IF M>=N THEN GOTO 250
230 LET B=0
240 GOTO 450
250 LET B=1
260 FOR I=M-N+1 TO M
270 LET B=B*I
280 NEXT I
290 GOTO 450
300 LET C$="COMB("
310 GOSUB 480
320 IF M>=N THEN GOTO 350
330 LET B=0
340 GOTO 450
350 IF M<>N THEN GOTO 380
360 LET B=1
370 GOTO 450
380 LET Y=N
390 IF M<2*N THEN LET Y=M-N
400 LET K=M+1
410 LET B=1
420 FOR I=1 TO Y
430 LET B=B*(K/I-1)
440 NEXT I
450 PRINT C$;M;",";N;")=";B
470 GOTO 30
480 PRINT "M=?";
490 INPUT M
495 PRINT M
500 PRINT "N=?";
510 INPUT N
515 PRINT N
520 IF M=INT ABS M AND N=INT ABS N AND M*N<>0
    THEN RETURN
530 PRINT "ERROR"
540 GOTO 480
```

12. Error Function ERF and Its Complement ERFC

PROGRAM DESCRIPTION

Attributes of many objects in nature, for example the height in inches of the male population of a certain city or the population IQs, follow closely the Normal or Gaussian distribution—the familiar bell curve. A particular Normal distribution is fully described by two parameters, the mean and the standard deviation. The Error Function (ERF) is equal to the cumulative Normal distribution and the Complementary Error Function (ERFC) equals to 1 – ERF according to the following definitions:

Formula 12-1

$$\mathrm{Erf}\,(\mu,\sigma,x) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt \qquad \mathrm{Erfc} = 1 - \mathrm{Erf}$$

The above formulas cannot be evaluated in closed form. The method used in this program to compute ERF and ERFC is iterative and leads to results that are correct within the full display capability of the Timex/Sinclair 1000 for both functions. By comparison, the standard method used to compute ERF and ERFC is based on polynomial expansion of a fifth or sixth degree and is precise to only 3-4 decimal places. ERFC computed via polynomial expansion completely loses accuracy at the tails of the distribution.

INSTRUCTIONS

Pressing the RUN key issues prompts for the distribution mean, standard deviation, and the upper distribution limit at which ERF and ERFC should be evaluated.

EXAMPLES

Problem Definition

1. Given a population with a mean IQ of 100 and a standard deviation of 15, what fraction of the population will have IQs larger than 135?
2. The average weight of a bag of fertilizer is 45-lbs. The manufacturing process introduces a standard deviation of 1.5-lbs. What ratio of bags will be heavier than 49.5-lbs?
3. You have probably heard about the 2.33 sigma limit (sigma is the standard deviation). What percentage of population does it include?

Sample Run

```
ERROR FUNCTION PROGRAM
DISTR. MEAN=?100
ST. DEVIATION=?15
UPPER DISTR. LIMIT=?135
ERF=0.99018467   ERFC=.0098153287

DISTR. MEAN=?45
ST. DEVIATION=?1.5
UPPER DISTR. LIMIT=?49.5
ERF=0.9986501    ERFC=.001349898

DISTR. MEAN=?100
ST. DEVIATION=?10
UPPER DISTR. LIMIT=?123.3
ERF=0.99009693   ERFC=.0099030756
```

Discussion of Results

Only 0.98% of the population will have IQs larger than 135. The second example shows that the danger of heavy bags is rather small, only 0.13%. The third example shows that 2.33 standard deviations include 99% of the population.

PROGRAMMING REMARKS

Depending on the value of the argument, the functions are evaluated as continuing fractions with a fixed number of elements or are evaluated iteratively until they converge.

PROGRAM LISTING

```
 20 CLS
 30 PRINT
 35 PRINT "ERROR FUNCTION PROGRAM"
 40 PRINT "DISTR. MEAN=?";
 50 INPUT B
 60 PRINT B
 70 PRINT "ST. DEVIATION=?";
 80 INPUT C
 90 PRINT C
100 IF C>=0 THEN GOTO 130
110 PRINT "ERROR"
120 GOTO 70
130 PRINT "UPPER DISTR. LIMIT=?";
140 INPUT D
150 PRINT D
160 LET X=(D-B)/C
170 LET X=X/SQR 2
180 LET Y=1/SQR PI*EXP (-X*X)
190 IF X<SQR 2 THEN GOTO 280
200 LET A=14/X
210 FOR I=27 TO 1 STEP -1
220 LET A=I/2/(X+A)
230 NEXT I
240 LET Z=Y/(X+A)
250 LET Z=Z/2
260 LET N=1-Z
270 GOTO 390
280 LET I=1
290 LET T=2*X*Y
300 LET N=T
310 LET T=2*X*X*T/(2*I+1)
320 LET M=N+T
330 IF M=N THEN GOTO 370
340 LET N=M
350 LET I=I+1
360 GOTO 310
370 LET N=N/2+.5
380 LET Z=1-N
390 PRINT "ERF=";N,"ERFC=";Z
400 PRINT
410 GOTO 40
```

13. Binomial and Hypergeometric Distributions

PROGRAM DESCRIPTION

These two distributions occur in many problems involving sampling theory, quality control, and games. The binomial distribution applies to the case of a large universe, while the hypergeometric distribution applies to cases of a small universe, where the samples are taken with no replacement. For example, the probability of finding X defective items in a sample from a large universe with a known proportion of defective items, would follow the binomial distribution. Similarly, the probability of finding exactly two red balls in a sample of five balls taken all at once from a universe of ten balls, three of which are red, would follow the hypergeometric distribution. Both distributions are very useful for estimating real life events. The following formulas apply for the two distributions:

A. Binomial distribution.

Given a large universe with a proportion of P special items (defective, marked, etc.) and a sample size of N, the probability of finding X special items in that sample is as follows:

Formula 13-1

$$\text{Prob (N,P,X)} = \binom{N}{X} p^{x} (1 - p)^{N-X}$$

B. Hypergeometric distribution.
Given a small universe of size M with K defective items, the probability of finding X defective items in a sample of size N is as follows:

Formula 13-2

$$\text{Prob (M,N,K,X)} = \frac{\binom{K}{X}\binom{M-K}{N-X}}{\binom{M}{N}}$$

INSTRUCTIONS

The first prompt after pressing the RUN key gives the choice of the binomial (1) or hypergeometric distribution (2). If the binomial distribution is selected, there follows a prompt for the sample size N and for the proportion of special items P in the universe. In this and following examples, when the prompt calls for more than one input value, key in each value separately followed by "Enter." The next prompt is for the number of special items X in the sample for which the probability should be computed. After displaying PROB (X) and cumulative probability, the next prompt gives a choice between a new X or a new run.

Similarly, for the hypergeometric distribution, the prompting sequence is the same, except that input of three parameters M, K, and N (size of the universe, number of special items and the sample size) is requested to describe the distribution.

EXAMPLES

Problem Definition

1. **Binomial distribution.** It is known that 4% of a manufacturer's brand are defective. What is the probability of finding at most one defective unit in a sample of five?
2. **Hypergeometric distribution.** In a box of 20 chocolates, five are dark, 15 are light. What is the probability of finding one or more dark chocolates in a handful of three?

Sample Run

```
BINOMIAL 1,HYPERGEOM. 2?1
ENTER N,P?5        .04
X=?1
PROB( 1 )=0.16986931
SUM OF P( X )=     0.16986931
NEW X 1,NEW RUN 2?1
X=?0
PROB( 0 )=0.8153727
SUM OF P( X )=     0.98524201
NEW X 1,NEW RUN 2?2

BINOMIAL 1,HYPERGEOM. 2?2
ENTER M,K,N        20
5                  3
X=?1
PROB( 20,5,3,1 )= 0.46052632
SUM OF( X )=0.46052632
```

```
NEW X 1,NEW DISTR. 2?1
X=?2
PROB( 20,5,3,2 )= 0.13157895
SUM OF( X )=0.59210526
NEW X 1,NEW DISTR. 2?1
X=?3
PROB( 20,5,3,3 )= .0087719298
SUM OF( X )=0.60087719
NEW X 1,NEW DISTR. 2?1
X=?0
PROB( 20,5,3,0 )= 0.39912281
SUM OF( X )=1
NEW X 1,NEW DISTR. 2?
```

Discussion of Results

In the first example, the probability of finding one defective unit in a sample is 0.169.., and of finding no defects is 0.815... The probability of finding 0 or 1 defective items in a sample of 5 is therefore P(0) + P(1) = 0.985...

In the second example the probability of finding one or more dark chocolates in a sample of size three is the sum of probabilities of finding one, two, or three, and is equal to 0.6. The same result can more easily be obtained by computing the probability of no dark chocolates and subtracting it from 1.0, as shown in the last run of the example.

PROGRAMMING REMARKS

Extensive checking of input values is performed in lines 90, 150, 190, 280, 360, and 400. Binomial coefficients are computed in the subroutine between lines 520 and 620 in the same way as in Program 11. Note use of PRINT statements in lines 230 and 460, which includes reprinting of input parameters for easier interpretation of results.

PROGRAM LISTING

```
20 CLS
40 PRINT
41 PRINT "BINOMIAL 1,HYPERGEOM. 2?";
45 LET Z=0
50 INPUT W
60 PRINT W
70 IF W=1 THEN GOTO 100
80 IF W=2 THEN GOTO 290
```

```
 90 GOTO 40
100 PRINT "ENTER N,P?";
110 INPUT N
120 PRINT N,
130 INPUT P
140 PRINT P
150 IF N<>INT N OR N<=0 OR P<=0 OR P>=1
    THEN GOTO 100
160 PRINT "X=?";
170 INPUT X
180 PRINT X
190 IF X<>INT X OR X<0 OR X>N THEN GOTO 160
200 GOSUB 520
210 LET T=B*P**X*(1-P)**(N-X)
220 LET Z=Z+T
230 PRINT "PROB(";X;")=";T,"SUM OF P(X)=",Z
240 PRINT "NEW X 1,NEW RUN 2?";
250 INPUT W
260 PRINT W
270 IF W=1 THEN GOTO 160
280 GOTO 40
290 PRINT "ENTER M,K,N",
300 INPUT E
310 PRINT E,
320 INPUT G
330 PRINT G,
340 INPUT F
350 PRINT F
360 IF E<>INT E OR G<>INT G OR F<>INT F
    THEN GOTO 290
365 IF E<=0 OR F<=0 OR G<=0 OR F>E OR G>E
    THEN GOTO 290
370 PRINT "X=?";
380 INPUT D
390 PRINT D
400 IF D<>INT D OR D<0 OR D>G THEN GOTO 370
410 LET N=G
420 LET X=D
422 GOSUB 520
424 LET J=B
426 LET N=E-G
428 LET X=F-D
430 GOSUB 520
432 LET J=J*B
```

```
434 LET N=E
436 LET X=F
438 GOSUB 520
440 LET J=J/B
450 LET Z=Z+J
460 PRINT "PROB(";E;",";G;",";
465 PRINT F;",";D;")=",J,"SUM OF(X)=";Z
470 PRINT "NEW X 1,NEW DISTR. 2?";
480 INPUT W
490 PRINT W
500 IF W=1 THEN GOTO 370
510 GOTO 40
520 LET Y=X
530 IF X>(N-X) THEN LET Y=N-X
540 IF Y<>0 THEN GOTO 570
550 LET B=1
560 RETURN
570 LET B=1
580 LET L=N+1
590 FOR I=1 TO Y
600 LET B=B*(L/I-1)
610 NEXT I
620 RETURN
```

14. Uniform and Normal (Gaussian) Pseudo-Random Deviates

PROGRAM DESCRIPTION

This program produces a sequence of either uniform or normally distributed (Gaussian) random deviates. Such sequences are frequently used for modeling with Monte Carlo method where a direct computation is not feasible.

An example of an application that is not amenable to exact calculation is that of telephone customer satisfaction. The satisfaction of a group of telephone subscribers depends on the probability distribution of talker loudness, distribution of noise, and distribution of transmission losses on a telephone connection. Though each use of these distributions can be estimated as Gaussian with a mean and standard deviation known from surveys, finding the combined distribution is computationally difficult. However, the combined effect of all these factors can be easily simulated by repeatedly generating pseudo-random deviates from each of these distributions and multiplying the individual probabilities.

This program would normally function as a subroutine in a user-provided application. The formulas used to generate the uniform and normal pseudo-random deviates are as follows:

A. Uniform Distribution

Formula 14-1

$$X_i = \text{Fract}\,(997\,X_{i-1} + \pi)$$

where RND is the Timex/Sinclair 1000 BASIC pseudo-random function, and X and Y are the lower and upper bounds of the random variable. The above formula generates uniform distribution values between the lower and upper bounds X and Y.

B. Normal (Gaussian) Distribution

Formula 14-2

$$X_i = (-\,2\ln U_1)^{1/2} \sin\,(2\pi\, U_2)$$

where U_1 and U_2 are two sequentially generated *uniformly* distributed pseudo-random deviates between 0 and 1. The resulting *normally* distributed pseudo-random deviates have a mean of 0 and a standard deviation of 1. The values are then transformed by the program into a distribution with an arbitrary mean M and standard deviation S by multiplying each random deviate with S and then adding M.

INSTRUCTIONS

Press the RUN key to start the program, then follow prompts for uniform (1) or Gaussian (2) deviates. The next prompt is for the starter which initializes the Timex/Sinclair 1000 RND function. The starter value should be between 0 and 65535. Entering 0 will always start a different random sequence; otherwise the sequence is predetermined. Then follow prompts for the lower and upper bounds for the uniform distribution, or for the mean and standard deviation of the Gaussian distribution, respectively.

EXAMPLES

Problem Definition

1. Generate a number of uniformly distributed deviates between 10 and 20.
2. Generate Gaussian deviates with mean of 100 and standard deviation of 10.

A possible interpretation of the resulting values of the Gaussian deviates could be a random reading of IQs.

Sample Run

```
UNIFORM 1, GAUSSIAN 2?1
STARTER=?123
LOWER, UPPER BOUND?10,20
RANDOM DEVIATE=11.418915

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=16.428375

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=12.132111

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=19.917297

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=13.797302
```

```
MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=14.804688

MORE VALUES 1, NEW RUN 2?2

UNIFORM 1, GAUSSIAN 2?2
STARTER=?555
MEAN=?100
ST. DEV.=?10
RANDOM DEVIATE=90.646835

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=114.96507

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=108.74587

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=116.17717

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=100.46652

MORE VALUES 1, NEW RUN 2?1
RANDOM DEVIATE=114.59181
```

Discussion of Results

The uniform deviates are between 10 and 20, the Gaussian deviates follow the normal distribution.

PROGRAMMING REMARKS

Error checking of all inputs is performed in lines 130, 220, and 280. The program shows the use of the BASIC RAND and RND functions. If the starter value, or argument, of RAND is selected as 0 then a new random sequence will be started every time the program is run. Otherwise the pseudo-random values can be repeated, if the same starter is chosen, in successive runs.

PROGRAM LISTING

```
20 CLS
25 PRINT
```

```
 30 PRINT "UNIFORM 1, GAUSSIAN 2?";
 40 INPUT B
 42 PRINT B
 44 PRINT "STARTER=?";
 46 INPUT S
 48 PRINT S
 50 RAND (S)
 52 IF B=1 THEN GOTO 80
 60 IF B=2 THEN GOTO 230
 70 GOTO 30
 80 PRINT "LOWER, UPPER BOUND?";
 90 INPUT X
100 PRINT X;
110 INPUT Y
120 PRINT ",";Y
130 IF Y>X THEN GOTO 160
140 PRINT "MIN>=MAX"
150 GOTO 80
160 PRINT "RANDOM DEVIATE=";RND*( Y-X )+X
165 PRINT
170 PRINT "MORE VALUES 1, NEW RUN 2?";
180 INPUT Z
190 PRINT Z
200 IF Z=1 THEN GOTO 160
210 IF Z=2 THEN GOTO 25
220 GOTO 170
230 PRINT "MEAN=?";
240 INPUT C
250 PRINT C
255 PRINT "ST. DEV.=?";
260 INPUT D
270 PRINT D
280 IF D>=0 THEN GOTO 310
290 PRINT "ST. DEV. < 0"
300 GOTO 255
310 PRINT "RANDOM DEVIATE=";
315 PRINT SQR (-2*LN RND)*SIN (2*PI*RND)*D+C
320 PRINT
325 PRINT "MORE VALUES 1, NEW RUN 2?";
330 INPUT Z
340 PRINT Z
350 IF Z=1 THEN GOTO 310
360 IF Z=2 THEN GOTO 25
370 GOTO 320
```

MATHEMATICS

15. Fourier Coefficients of Periodic Functions

PROGRAM DESCRIPTION

A continuous periodic function $y = f(x)$ with a period T defined between $-T/2$ and $+T/2$ can be represented as the sum of sine and cosine terms according to the following classical formulas attributed to the French mathematician Fourier:

Formula 15-1

$$f(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty} (A_n \cos 2n \frac{\pi}{T} x + B_n \sin 2n \frac{\pi}{T} x) = \frac{A_0}{2} + \sum_{n=1}^{\infty} C_n \cos (2n \frac{\pi}{T} x + \theta)$$

$$A_n = \frac{2}{T} \int_{-T/2}^{T/2} f(x) \cos \frac{2n \pi x}{T} dx \qquad C_n = \sqrt{A_n^2 + B_n^2}$$

$$B_n = \frac{2}{T} \int_{-T/2}^{T/2} f(x) \sin \frac{2n \pi x}{T} dx \qquad \theta = \text{Arctan} (- \frac{Bn}{An})$$

This program evaluates the above expressions by performing numerical integration at specified sampling intervals. If it is known in advance that the function is even, $f(x) = f(-x)$, or that it is odd, $f(x) = -f(-x)$, then the program can be speeded up because only A or B coefficients have to be evaluated and the integration can proceed over only one half of the period T. The program takes advantage of this information.

Though the program only displays the successive Fourier coefficients, these coefficients could easily be stored at the time of computation in a dimensioned variable (memory size permitting) for future use.

INSTRUCTIONS

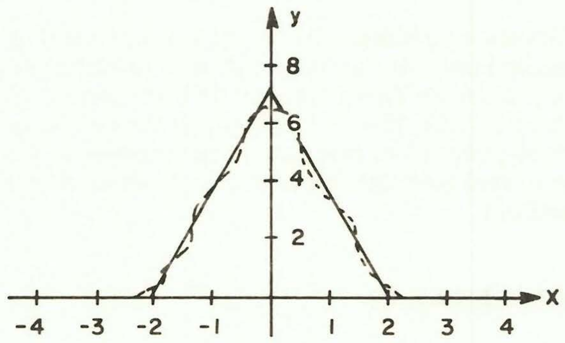
Before running the program, write a subroutine defining the function to be evaluated starting on line 1000 and ending with a RETURN (see program listing and example). Pressing the RUN key initializes the program and issues a prompt for the type of function: (1) for an even function, (2) for an odd function, and (3) if the type of function is not known. The subsequent prompts request the size of the period T and the number of sampling intervals (two or more). Selection of a large number of intervals results in greater accuracy but increased computation time. For an even or odd function the requested number of intervals applies to only one half of the period T, thus making the intervals smaller and the accuracy of the computation better. When a Fourier coefficient is found the computer will flash its value. Successive coefficients are computed and displayed until the screen is full (error 5) or the BREAK key is pressed.

EXAMPLE

Problem Definition

The function specified in lines 1000–1020 is shown in Figure 15-1. Its period is between −3.5 and +3.5. The program computes successive Fourier coefficients A0, A1, A2. . . . for this even function of x. The approximated curve is shown dashed in the same figure. The computation is to be performed with 10 and 100 intervals.

Figure 15-1



Sample Run

```
EVEN FUNCT. 1,ODD 2,
NEITHER 3?1
PERIOD T=?7
NO. INTERVALS=?10
A0=4.025
A1=3.0177238
A2=1.1368327
A3=0.1117739
A4=.070788988
A5=0.14849242
A6=.058580645
A7=.01478557
A8=.027038986
A9=.011989939
A10=0
A11=-.011989941
A12=-.027038986
A13=-.014785568
A14=-.058580644
A15=-0.14849242
A16=-.070788986
A17=-0.1117739
```

```
EVEN FUNCT. 1,ODD 2,
NEITHER 3?1
PERIOD T=?7
NO. INTERVALS=?100
A0=4.00015
A1=3.0345888
A2=1.1793983
A3=0.10390619
A4=.058466219
A5=0.18842451
A6=.08414665
A7=.0001493158
A8=.047250562
A9=.057932911
A10=.0094089601
A11=.0077683475
A12=.03243459
A13=.017811636
A14=.00014726251
A15=.013311727
A16=.018110545
```

Discussion of Results

The Fourier coefficients were independently verified with a high precision algorithm on a different computer; it was found that with 100 intervals between four and five decimal places on the Timex/Sinclair 1000 are correct. Note the Gibbs phenomenon at sharp corners, $X = -3.5, 0, +3.5$; the Fourier expansion does not converge at those points to its true value. For example, at $X = 0$ the series seems to converge toward 6.85 rather than 7.0 as expected. At other points the convergence is excellent.

PROGRAMMING REMARKS

Checks on input errors are performed in lines 60, 100, and 140. The program performs trapezoidal integration of the function starting in line 1000.

PROGRAM LISTING

```
 30 PRINT "EVEN FUNCT. 1,ODD 2,NEITHER 3?";
 40 INPUT A
 50 PRINT A
 60 IF A<1 OR A>3 OR A<>INT A THEN GOTO 30
 70 PRINT "PERIOD T=?";
 80 INPUT B
 90 PRINT B
100 IF B<=0 THEN GOTO 70
110 PRINT "NO. INTERVALS=?";
120 INPUT C
130 PRINT C
140 IF C<2 OR C<>INT C THEN GOTO 110
150 IF A=3 THEN GOTO 500
160 LET D=B/2/C
170 IF A=2 THEN GOTO 340
180 LET J=-1
190 LET J=J+1
195 PAUSE 20
200 LET X=0
210 GOSUB 1000
220 LET E=Y
230 LET F=0
240 LET G=2*J*PI/B
250 FOR I=1 TO C
260 LET X=D*I
270 GOSUB 1000
280 LET F=F+(E+Y)/2*D*COS (G*(X-D/2))
290 LET E=Y
300 NEXT I
310 LET H=4*F/B
320 PRINT "A";J;"=";H
330 GOTO 190
340 LET J=0
350 LET J=J+1
355 PAUSE 20
360 LET X=0
370 GOSUB 1000
380 LET E=Y
390 LET K=0
400 LET G=2*J*PI/B
410 FOR I=1 TO C
```

```
 420 LET X=D*I
 430 GOSUB 1000
 440 LET K=K+(E+Y)/2*D*SIN (G*(X-D/2))
 450 LET E=Y
 460 NEXT I
 470 LET L=4*K/B
 480 PRINT "B";J;"=";L
 490 GOTO 350
 500 LET D=B/C
 510 LET J=-1
 520 LET J=J+1
 525 PAUSE 20
 530 LET X=-B/2
 540 GOSUB 1000
 550 LET E=Y
 560 LET K=0
 570 LET F=0
 580 LET G=2*J*PI/B
 590 FOR I=1 TO C
 600 LET X=D*I-B/2
 610 GOSUB 1000
 620 LET M=G*(X-D/2)
 630 LET F=F+(E+Y)/2*D*COS M
 640 IF J=0 THEN GOTO 660
 650 LET K=K+(E+Y)/2*D*SIN M
 660 LET E=Y
 670 NEXT I
 680 LET H=2*F/B
 690 PRINT "A";J;"=";H
 700 IF J=0 THEN GOTO 520
 710 LET L=2*K/B
 720 PRINT "B";J;"=";L
 730 LET N=SQR (H*H+L*L)
 740 LET P=ATN (-L/H)
 750 PRINT "C";J;"=";N
 760 PRINT "THETA";J;"=";P
 770 GOTO 520
1000 LET Y=7-3.5*ABS X
1010 IF ABS X>2 THEN LET Y=0
1020 RETURN
```

16. Runge-Kutta Solutions of Differential Equations

PROGRAM DESCRIPTION

Runge-Kutta methods for numerical solution of ordinary differential equations consist of a group of iterative formulas distinguished by certain important common features:

1. The derivative y′ can be given as a function of x and y. It does not have to be provided as an explicit function of x alone.
2. The method is self-starting. Knowledge of the function is only required at one point to compute the succeeding point in the iteration.
3. The integration interval can be changed as the iteration proceeds.
4. The formulas usually converge quickly.

Successively higher order Runge-Kutta formulas and finer integration intervals lead to more precise solutions. The program evaluates fourth order Runge-Kutta formulas for solving first order differential equations of the type:

$$y' = f(x,y)$$

The formulas assume that f(x,y) is known. The initial condition X0 and Y0, the integration interval H along the X-axis, and the final value of X are selected by the user. The program evaluates the following formulas:

Formula 16-1

$$K_1 = Hf(X_i,\ Y_i)$$

$$K_2 = Hf(X_i + H/2,\ Y_i + K_1/2)$$

$$K_3 = Hf(X_i + H/2,\ Y_i + K_2/2)$$

$$K_4 \Rightarrow Hf(X_i + H/2,\ Y_i + K_3/2)$$

$$Y_{i+1} = Y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

INSTRUCTIONS

Before running the program write the function y′ = f(x,y) as a BASIC subroutine starting on line 1000 and append it to the program as shown in the program listing and in the example. Use the letter R for the derivative. Pressing the RUN key starts the program and prompts for the interval size H, the initial values of X0 and Y0, and for the final value of X at which the solution should be found. The interval H should be positive, though X can be smaller or larger than X0. There are no other limitations on H. The program flashes the succeeding x values as they are evaluated and stops when the computation is completed by displaying Y(X).

EXAMPLE

Problem Definition

Given the differential equation y′ = x + y, with the initial condition X= 0, Y = 0. Find Y(1). Assume integration intervals of 0.15 and 0.05.

Sample Run

```
4-TH ORDER RUNGE KUTTA
INTEGRATION INTERVAL H?0.15
ENTER X0,Y0,XLAST?0,0,1
0.15,0.3,0.45,0.6,0.75,0.9,1,
Y( 1 )=0.71827251

INTEGRATION INTERVAL H?.05
ENTER X0,Y0,XLAST?0,0,1
.05,0.1,0.15,0.2,0.25,0.3,0.35,0
.4,0.45,0.5,0.55,0.6,0.65,0.7,0.
75,0.8,0.85,0.9,0.95,1,1,
Y( 1 )=0.71828169
```

Discussion of Results

This happens to be a differential equation for which an exact solution can be found. The solution is:

Y(1) = e − 2 = 0.718281...

The Runge-Kutta solution is 0.718272... for H = 0.15 and 0.718281 for the integration interval of 0.05. The shorter integration interval evidently produces a more exact solution.

PROGRAMMING REMARKS

Input error check on the interval size is performed in line 60. Notice the display of the final value of Y(X) in line 245. The statement uses XLAST as its argument for clarity. As mentioned in the Program Description, the routine starting in line 1000 describes the particular differential equation to be evaluated.

PROGRAM LISTING

```
 20 PRINT "4-TH ORDER RUNGE KUTTA"
 30 PRINT "INTEGRATION INTERVAL H?";
 40 INPUT A
 50 PRINT A
 60 IF A<=0 THEN GOTO 30
 70 PRINT "ENTER X0,Y0,XLAST?";
 80 INPUT B
 90 PRINT B;
100 INPUT C
110 PRINT ",";C;
120 INPUT D
130 PRINT ",";D
135 IF B=D THEN GOTO 70
140 LET E=INT ABS (D-B)/A
150 LET F=B
160 LET G=SGN (D-B)*A
170 LET H=C
180 FOR I=1 TO E
190 GOSUB 270
200 NEXT I
210 IF F=D THEN GOTO 240
220 LET G=D-F
230 GOSUB 270
240 PRINT
245 PRINT "Y( ";D;" )=";H
250 PRINT
260 GOTO 30
270 LET X=F
280 LET Y=H
290 GOSUB 1000
300 LET K=G*R
310 LET X=F+G/2
320 LET Y=H+K/2
```

```
 330 GOSUB 1000
 340 LET L=G*R
 350 LET Y=H+L/2
 360 GOSUB 1000
 370 LET M=G*R
 380 LET X=F+G
 390 LET Y=H+M
 400 GOSUB 1000
 410 LET N=G*R
 420 LET H=H+K/6+L/3+M/3+N/6
 430 LET F=F+G
 440 PRINT F;",";
 450 PAUSE 20
 460 RETURN
1000 LET R=X+Y
1010 RETURN
```

17. Numeric Solution of Transcendental Equations

PROGRAM DESCRIPTION

This program solves numerical equations which in general cannot easily be solved explicitly. For example, it will solve for X:

$F(X) = LOG\,X - X^2 + 10 = 0.$

The program requests an initial estimate of X0 in the neighborhood of the solution. Such an estimate can usually be made from a plot of the function. It then requests a DELTA increment at which the function will be evaluated (a large DELTA will converge more quickly, but may miss the true solution). The program then requests EPSILON, a small positive number. When the absolute value of the function is less than EPSILON, iteration stops and the solution is printed. If more than one solution exists, then the program will find the solution closest to the initial estimate.

The program uses a modified Newton's iteration method to find the zero crossing of the function, which is the required solution. The program starts by "drawing" a straight line through the function evaluated at X0 + DELTA/2 and X0 − DELTA/2. The crossing of this line with the X axis gives the next estimate of X0. If the estimate of X0 stops changing while the absolute value of F(X) is still larger than EPSILON, then the DELTA interval is halved by the program and the iteration continues. It is obvious that this method will only work for functions without irregularities between the first estimate of X0 and the true X0. The function F(X) also has to be single-valued and its first derivative cannot be 0 in this region, otherwise the straight horizontal line will not cross the X-axis. To indicate how the iteration is progressing (hopefully it is converging!), the program flashes successive approximations of X0. As long as these are converging you should arrive at the approximate solution within a short time. The program then stops and displays its best estimate of the solution to the equation.

INSTRUCTIONS

As in the previous program, the equation to be solved is appended to the program before running it. The subroutine describing the equation should start at line 1000 and should end with a RETURN as shown in the listing and example. Pressing the RUN key starts the program and generates three prompts, for the initial guess of X0, DELTA, and EPSILON. The iteration then continues until ABS (F(X)) < EPSILON.

EXAMPLE

Problem Definition

We will try to solve the equation mentioned earlier:

$F(X) = \mathrm{LOG}\,X - X^2 + 10 = 0.$

Use the initial guess for X0 = 2.0, DELTA = 0.01, and EPSILON = 1E−6, then repeat with a larger DELTA of 0.1.

Sample Run

```
TRANSCENDENTAL EQUATION SOLUTION
ENTER GUESS FOR X0,DELTA X,EPS.?
2,.01,1E-6
SUCCESSIVE APPROXIMATIONS:
3.6656754          3.2673533
3.2421374          3.2420352
3.2420352          3.2420352
3.2420352          3.2420382
3.2420382          3.242039
3.242039           3.2420392

X0=3.2420392
```

```
TRANSCENDENTAL EQUATION SOLUTION
ENTER GUESS FOR X0,DELTA X,EPS.?
2,0.1,1E-6
SUCCESSIVE APPROXIMATIONS:
3.6650052          3.266929
3.2417394          3.2416374
3.2416374          3.2416374
3.2419388          3.2419388
3.2419388          3.2420141
3.2420141          3.2420141
3.2420329          3.2420329
3.2420377          3.2420377
3.2420388          3.2420388
3.2420391

X0=3.2420391
```

Problem Discussion

Both runs, after happily flashing converging values of X, lead to the result of X=3.242...

PROGRAMMING REMARKS

The program flow ends with a STOP in line 260 in order not to collide with the subroutine describing the equation. For every run you must press the RUN key.

PROGRAM LISTING

```
  10 PRINT "TRANSCENDENTAL EQUATION SOLUTION"
  20 PRINT "ENTER GUESS FOR X0,DELTA X,EPS.?"
  30 INPUT A
  35 LET E=A
  40 PRINT A;
  50 INPUT B
  60 PRINT ",";B;
  70 INPUT C
  80 PRINT ",";C
  85 PRINT "SUCCESSIVE APPROXIMATIONS:"
  90 LET X=A-B/2
 100 GOSUB 1000
 110 LET D=Y
 120 LET X=A+B/2
 130 GOSUB 1000
 140 LET Z=B*D/(D-Y)+A-B/2
 150 LET X=Z
 160 PRINT Z,
 170 PAUSE 30
 180 GOSUB 1000
 190 IF ABS Y<=ABS C THEN GOTO 240
 200 IF Y=E THEN LET B=B/2
 210 LET A=Z
 220 LET E=Y
 230 GOTO 90
 240 PRINT
 245 PRINT
 250 PRINT "X0=";Z
 260 STOP
1000 LET Y=LN X/LN 10-X*X+10
1010 RETURN
```

18. Complex Arithmetic

PROGRAM DESCRIPTION

With this program your Timex/Sinclair 1000 computer becomes a full-fledged complex arithmetic calculator operating in Reverse Polish Notation (RPN). For those of you familiar with Hewlett Packard calculators, which use RPN, this will be old hat. If you are unfamiliar with RPN a few explanations are necessary.

Think of a stack of registers, four deep, named X, Y, Z, and T in your computer. The bottom of the stack, the X register and the one immediately above it, the Y register, are always being displayed. Each register contains the real and the imaginery parts of a complex number. Unary (one variable) operations; e.g., squaring, taking the square root, computing reciprocals, etc. are performed on the X register alone and do not affect the stack. Operations on two variables, such as addition, subtraction, multiplication, and division operate on the X and Y registers. The result of a two-variable operation always appears in the X register and the stack drops, that is, the contents of register Z are moved into register Y, and the register T is copied into Z. Register T retains its previous value. Numbers are entered into the stack via the ENTER command (1). Entering a complex number moves it into the X register replacing the previous contents of that register. Several stack manipulation operations are provided: X<->Y exchanges the contents of the X and Y registers, RAISE STACK destroys the previous contents of register T, moves previous contents of register Z into T, Y into Z, and duplicates X into Y. The program also provides a memory register M. Two commands are provided to copy register X into M (SAVE), and to copy register M in X (RECALL).

INSTRUCTIONS

RUN — Start computation, issue prompts for operations 1-12
1 — ENTER, enters the complex number into X register, stack remains unchanged
2 — RAISE STACK, copies contents of Z in T, Y in Z, and X in Y; destroys former contents of T.
3 — X<->Y, exchanges the X and Y registers
4 — SAVE, copies X into M
5 — RECALL, copies M into X
6 — Adds X and Y, puts result in X, drops stack
7 — Subtracts X from Y, puts result in X, drops stack
8 — Multiplies X and Y, puts result in X, drops stack
9 — Divides Y by X, puts result in X, drops stack
10 — Finds reciprocal of X, puts it in X
11 — Finds square root of X, puts it in X
12 — Finds square of X, puts it in X

EXAMPLE

Problem Definition

Evaluate the following expression:

Formula 18-1

$$\frac{(2 + 3j)(5 - 4j)}{5 + 4j}$$

Sample Run

```
OPERATION 1-12?1
ENTER R,I?2,3
   REAL             IMAGINARY
Y: 0                0
X: 2                3

OPERATION 1-12?3
   REAL             IMAGINARY
Y: 2                3
X: 0                0

OPERATION 1-12?1
ENTER R,I?5,-4
   REAL             IMAGINARY
Y: 2                3
X: 5                -4

OPERATION 1-12?8
   REAL             IMAGINARY
Y: 0                0
X: 22               7

OPERATION 1-12?3
   REAL             IMAGINARY
Y: 22               7
X: 0                0
```

```
OPERATION 1-12?1
ENTER R,I?5,4
   REAL          IMAGINARY
Y: 22            7
X: 5             4

OPERATION 1-12?9
   REAL          IMAGINARY
Y: 0             0
X: 3.3658537     -1.2926829

OPERATION 1-12?11
   REAL          IMAGINARY
Y: 0             0
X: 1.8670036     -0.34619187

OPERATION 1-12?12
   REAL          IMAGINARY
Y: 0             0
X: 3.3658537     -1.2926829
```

Discussion of Results

The complex number 2 + 3j is entered first. It is then moved from the X into the Y register before the second complex number 5 − 4j is entered. X and Y are then multiplied and the intermediate result is 22 + 7j. The product is then moved into the Y register and the last complex number 5 + 4j is entered. Next a division and a square root operation is performed. The result is then 1.86. . . − 0.346. . . j. In the final step the result is squared to demonstrate that the complex square root and squaring are inverse operations. Note that a second root also exists which is the negative of the first one, namely − 1.86. . . + 0.346. . . j.

PROGRAMMING REMARKS

The program fills the 2K RAM capacity of the Timex/Sinclair 1000. Therefore only the X and Y registers are displayed rather than the whole stack. The square root function (11) displays only one root. The other root is the negative of the first one. Stack raising and dropping is performed in FOR NEXT loops in lines 100–110 and 1000–1015. The only input error check is performed in line 27. Multiple branching is performed in line 40 by multiplying the operation code by 50.

PROGRAM LISTING

```
 10 DIM X(8)
 15 PRINT
 18 PRINT "OPERATION 1-12?";
 20 INPUT F
 25 PRINT F
 27 IF F<>INT F OR F<1 OR F>12 THEN GOTO 18
 40 GOTO 50*F
 50 PRINT "ENTER R,I?";
 55 INPUT X(1)
 60 PRINT X(1);
 65 INPUT X(2)
 67 PRINT ",";X(2)
 70 LET R=X(1)
 72 LET J=X(2)
 73 PRINT "    REAL","IMAGINARY","Y:
    ";X(3),X(4),"X: ";R,J
 74 LET D=R*R+J*J
 75 GOTO 15
100 FOR I=8 TO 3 STEP -1
105 LET X(I)=X(I-2)
110 NEXT I
115 GOTO 70
150 LET X(1)=X(3)
165 LET X(2)=X(4)
170 LET X(3)=R
175 LET X(4)=J
180 GOTO 70
200 LET K=R
205 LET M=J
210 GOTO 70
250 LET X(1)=K
255 LET X(2)=M
260 GOTO 70
300 LET X(1)=R+X(3)
305 LET X(2)=J+X(4)
310 GOTO 1000
350 LET X(1)=X(3)-R
355 LET X(2)=X(4)-J
360 GOTO 1000
400 LET X(1)=R*X(3)-J*X(4)
405 LET X(2)=R*X(4)+J*X(3)
410 GOTO 1000
```

```
 450 LET X(1)=(R*X(3)+J*X(4))/D
 455 LET X(2)=(R*X(4)-J*X(3))/D
 460 GOTO 1000
 500 LET X(1)=R/D
 505 LET X(2)=-J/D
 510 GOTO 1000
 550 IF R<>0 OR J<>0 THEN GOTO 554
 551 LET X(1)=0
 552 LET X(2)=0
 553 GOTO 70
 554 LET X(1)=SQR ((R+SQR (R*R+J*J))/2)
 555 LET X(2)=J/2/X(1)
 560 GOTO 70
 600 LET X(1)=R*R-J*J
 605 LET X(2)=2*R*J
 610 GOTO 70
1000 FOR I=3 TO 6
1010 LET X(I)=X(I+2)
1015 NEXT I
1020 GOTO 70
```

19. Decimal-to-Binary Fractions

PROGRAM DESCRIPTION

That part of the civilized world that employs the British system of measurements still prefers binary fractions such as 5/16 or 7/32 to numbers as easy to remember as 0.3125 or 0.21875. To accommodate both the "decimal" and the "binary" population, this program will translate a decimal fraction into its binary equivalent. We will define a "binary" fraction as a fraction with a denominator that is an integer power of 2. All the program user has to do is to indicate the smallest acceptable fraction—a binary power of 2; e.g., 256 for 1/256, 64 for 1/64, etc.

If the translation can be done exactly, then the program will divide the numerator by the denominator. For example, 0.375 will be translated to 3/8 though the smallest fraction requested by the user may be 1/256. If the translation cannot be done exactly, the smallest binary denominator will be found within the given constraints, and the percent error will be indicated.

INSTRUCTIONS

Pressing the RUN key issues two prompts: ENTER DECIMAL FRACTION?—the number to be translated into the binary fraction, and MAX DENOMINATOR (PWR OF 2)—the number to indicate the largest acceptable denominator. The program then displays the binary fraction and the error, if any.

EXAMPLE

Problem Definition

Translate 2.5, 0.12717, and 0.1875 into binary fractions. Do not accept any fraction smaller than 1/16 for the first example and 1/256 for the second and third examples.

Sample Run

```
ENTER DECIMAL FRACTION?2.5
MAX. DENOMINATOR (PWR OF 2)?16
2.5=5/2

ENTER DECIMAL FRACTION?0.12717
MAX. DENOMINATOR (PWR OF 2)?256
0.12717 APPROXIMATES 33/256
WITH ERROR OF -1.3652984 PERCENT

ENTER DECIMAL FRACTION?0.1875
MAX. DENOMINATOR (PWR OF 2)?256
0.1875=3/16
```

Discussion of Results

The first and third examples lead to exact binary fractions. The second example can be approximated wtihin the constraints of 1/256 to within 1.35. . . %.

PROGRAMMING REMARKS

The only error check on input value (largest denominator) is performed in lines 70–90. To account for the Timex/Sinclair 1000 roundoff a "fudge" factor of 0.0001 is used in lines 90 and 120. To find the smallest denominator the program keeps dividing the numerator and the denominator by 2 (lines 140–150), as long as both are integers.

PROGRAM LISTING

```
 5 PRINT
10 PRINT "ENTER DECIMAL FRACTION?";
20 INPUT A
30 PRINT A
40 PRINT "MAX. DENOMINATOR (PWR OF 2)?";
50 INPUT B
60 PRINT B
70 IF B<2 THEN GOTO 40
80 LET C=LN B/LN 2
90 IF ABS (C-INT C)>.0001 THEN GOTO 40
```

```
100 LET C=INT C
110 LET D=A*B
120 IF (ABS D-INT ABS (D+.0001)>=.0001)
    THEN GOTO 170
130 IF D/2<>INT (D/2) THEN GOTO 230
140 LET D=D/2
150 LET B=B/2
160 GOTO 130
170 LET D=INT (D+.5)
180 LET E=A-D/B
190 LET F=100*E/A
200 PRINT A;" APPROXIMATES ";D;"/";B
210 PRINT "WITH ERROR OF ";F;" PERCENT"
220 GOTO 5
230 PRINT A;"=";D;"/";B
240 GOTO 5
```

OPERATIONS RESEARCH

20. Solution of Queuing Equations

PROGRAM DESCRIPTION

The program presented here solves the classical M/M/S queuing equations which describe many real life situations.

The M/M/S queuing equations assume exponential distribution of customer arrival rates L, exponential distribution of serving times M, S identical servers, and FIFO order of service. Exponential distribution implies independence of events; the customers will seek service independent of the queue length and the servers will operate at a steady rate independent of the load and queue length. Such assumptions usually result in safe (pessimistic) estimates of waiting times and other queuing parameters. The M/M/S cases result in relatively simple equations (in terms of queuingtheory) and can be solved within the memory limitations of the Timex/Sinclair 1000. The program can be used to estimate such problems as how many barber chairs to provide at a midtown location, how many telephone operators should be working between 2 and 3am, or how many printers should be connected to a central processor. In general, application of the program will help in balancing the cost of providing service with the inconvenience of long waiting lines. The queuing equations used in the program are given below.

Program limitations: $S < 34$, $U < = 1.0$, all inputs positive, all intermediate results should be less than 1.0E38.

Inputs:
Av. arrival rate r
Av. service rate m
Number of servers s
Number of items in system n

Outputs:
Facility utilization $U = r/(ms)$
Prob. of finding no items in system P(0)

$$P(0) = ((r/m)^s /s!(1-U) + \sum_{j=0}^{s-1} (r/m)^j /j!)^{-1}$$

Prob. of finding *n* items in the system P(n)
  $P(n) = P(0) \times (r/m)^n \times (1/n!)$ for $n<s$
  $P(n) = P(0) \times (r/m)^n \times (1/(s!s^{n-s}))$ for $n>=s$
Prob. of finding all servers busy B

$$B = \sum_{n=s}^{\infty} P(n) = (r/m)^s \times P(0)/(s! \times (1-U))$$

Average waiting in queue $Tw = B/(sm \times (1-U))$
Average system response time $Tr = Tw + 1/m$
St. dev. of $Tr = Tw/B \times \sqrt{(B(2-B) + s^2 \times (1-U)^2)}$
Prob. of $Tw>T = B \times Exp(-smT \times (1-U))$
Prob. of $Tr>T = Exp(-mT) \times (1 + (1 - Exp(-msTk)) \times B/sk)$ for $k <> 0$ where $k = 1 - U - 1/s$
Prob. of $Tr>T = Exp(-mT) \times (1 + BmT)$ for $k = 0$
Average number of items in queue $Q = r \times Tw$
Average number of items in system $N = r \times Tr$

The program provides all prompting and checks input data for correct values.

PROGRAM INSTRUCTIONS

Pressing the RUN key will issue prompts to enter the arrival rate L, the service rate M and the number of servers S. An incorrect input will cause the prompts to repeat. The program then computes and displays the previously mentioned parameters of the queuing system, namely B, U, P(0), Tw, Tr, Standard deviation of Tr, Q, and N. The program will then issue a prompt "ENTER CODE 1, 2 OR 3?" To compute P(n) enter 1, P(Tw,Tr)—2, or to continue with a new run—3. Selecting P(n) will elicit a prompt for the number of items in the system n and will compute and display P(n). Choosing P(Tw,Tr) will prompt for time T and will compute and display P(Tw > T) and P(Tr > T). The prompts then repeat.

EXAMPLE

Problem Definition

A computer working area has five terminals. On average 16 customers per hour arrive at random intervals with each job taking on average 15 minutes (four per hour). Compute the queuing parameters; also find the reduction in waiting time if a sixth terminal is installed.

Sample Run

```
ENTER L,M,S?16,4,5
P( ALL BUSY )=    0.55411255
UTIL=             0.8
P( 0 )=           .012987013
AV.WAIT=          0.13852814
AV.RESP.=         0.38852814
```

```
ST.DEV.(TR)=      0.33552053
AV. IN SYS.=      6.2164502
AV IN QUEUE=      2.2164502

ENTER CODE 1,2 OR 3?1
NO. ITEMS IN SYSTEM?1
P(N=1)=.051948052

ENTER CODE 1,2 OR 3?1
NO. ITEMS IN SYSTEM?6
P(N=6)=.088658009

ENTER CODE 1,2 OR 3?1
NO. ITEMS IN SYSTEM?10
P(N=10)=.03631432

ENTER CODE 1,2 OR 3?2
TIME=?0.2
P(TW>0.2)=0.24897882
P(TR>0.2)=0.64851202

ENTER CODE 1,2 OR 3?2
TIME=?10
P(TW>10)=2.3540663E-18
P(TR>10)=9.8411012E-17
```

```
ENTER L,M,S?16,4,6
P(ALL BUSY)= 0.28476085
UTIL=        0.66666667
P(0)=        .016685206
AV.WAIT=     .035595106
AV.RESP.=    0.28559511
ST.DEV.(TR)= 0.26482403
AV. IN SYS.= 4.5695217
AV IN QUEUE= 0.56952169

ENTER CODE 1,2 OR 3?1
NO. ITEMS IN SYSTEM?1
P(N=1)=.066740823

ENTER CODE 1,2 OR 3?2
TIME=?0.2
P(TW>0.2)=.057492223
P(TR>0.2)=0.51978804
```

Discussion of Results

The first part of the example (five terminals) shows a fairly high utilization of the facility, 0.8. The average waiting time in queue is 0.138 hours or slightly over eight minutes. Adding a sixth terminal will cause a considerable improvement. The waiting time will drop to 0.035 hours or about two minutes. All other wait probabilities are also correspondingly affected.

PROGRAMMING REMARKS

A check of input values is performed in line 80. Notice use of the roundoff fudge factor in line 60. A fairly elaborate PRINT statement is shown in line 900. As you enter the program notice the use of the EXP function (Shift ENTER to get in the FUNCTION mode, followed by pressing the X key).

PROGRAM LISTING

```
  5 CLS
 10 PRINT "ENTER L,M,S?";
 20 INPUT L
 30 PRINT L;
 40 INPUT M
 50 PRINT ",";M;
 60 INPUT S
 70 PRINT ",";S
 80 IF L<=0 OR M<=0 OR S<=0 OR S<>INT S OR S>33
    THEN GOTO 10
 85 LET P=0
 90 LET X=L/M
100 LET U=X/S
110 LET V=1-U
120 IF V<=0 THEN GOTO 10
130 LET Y=S
140 GOSUB 1000
150 LET T=Z
160 FOR I=0 TO S-1
170 LET Y=I
180 GOSUB 1000
190 LET P=P+X**I/Z
200 NEXT I
210 LET Y=S
220 GOSUB 1000
230 LET P=1/(P+X**S/T/V)
240 LET B=X**S*P/T/V
250 LET W=B/S/M/V
260 LET R=W+1/M
270 PRINT "P(ALL BUSY)=",B,"UTIL=",U,"P(0)=",
275 PRINT P,"AV.WAIT=",W,"AV.RESP.=",R
280 PRINT "ST.DEV.(TR)=",SQR (2*B-B*B+(S*V)**2)
    *W/B
290 PRINT "AV. IN SYS.=",L*R,"AV IN QUEUE=",L*W
350 PRINT
355 PRINT "ENTER CODE 1,2 OR 3?";
360 INPUT K
370 PRINT K
380 IF K<>1 THEN GOTO 510
390 PRINT "NO. ITEMS IN SYSTEM?";
400 INPUT N
410 PRINT N
420 LET D=P*X**N
```

```
 430 IF N<S THEN GOTO 460
 440 LET E=D/T/S**(N-S)
 450 GOTO 490
 460 LET Y=N
 470 GOSUB 1000
 480 LET E=D/Z
 490 PRINT "P(N=";N;")=";E
 500 GOTO 350
 510 IF K<>2 THEN GOTO 5
 520 PRINT "TIME=?";
 530 INPUT H
 540 PRINT H
 550 LET F=B*EXP -(S*M*V*H)
 560 LET A$="W"
 570 GOSUB 900
 580 LET A$="R"
 590 LET G=V-1/S
 600 LET D=EXP -(M*H)
 610 IF ABS G<=.00001 THEN GOTO 640
 620 LET F=D*(1+B/S*(1-EXP -(M*S*G*H))/G)
 630 GOTO 650
 640 LET F=D*(1+B*M*H)
 650 GOSUB 900
 660 GOTO 350
 900 PRINT "P(T";A$;">";H;")=";F
 910 RETURN
1000 LET Z=1
1010 FOR J=1 TO Y
1020 LET Z=Z*J
1030 NEXT J
1040 RETURN
```

21. Reliability Parameters of Complex Systems

PROGRAM DESCRIPTION

Reliability of a composite system; e.g., a telephone office consisting of incoming and outgoing trunks, switching machines, and power supplies or a computation center consisting of CPUs, printers, disk drives, and power supplies, can be expressed in terms of the Mean Time Before Failure (MTBF), and Mean Time Till Repair (MTTR). A high MTBF and a low MTTR can be achieved by using reliable components which do not break down frequently and are easy to repair, or by providing redundancy. For example, two CPU's can share the load; if one fails, the other takes over.

The program will find the overall reliability of a system given MTBFs and MTTRs of individual components. It is assumed that failures of individual components are independent of one another. The reliability curve then follows the center part of the classical "bathtub" curve, where the high number of failures in the beginning is caused by initial problems and the high number of failures towards the end of useful life is caused by component wearout. It is also assumed that individual components will be grouped into subsystems of one of three types listed below, and that a number of subsystems are connected in series to form an overall system.

A. Series subsystems. Failure of one component will incapacitate the whole system.

B. Parallel subsystems. At least one component has to stay alive for the system to operate.

C. Parallel k-of-n subsystem. Where k components out of n have to stay alive for the system to operate.

The program computes MTBF and MTTR for each subsystem and for the overall system. It also finds the availability factor and the probability of x components failing during time t, where x and t can be selected.

The following equations are used:

Formula 21-1

A–Availability
U–Unavailability
N–# of elements in subsystem
k–# of elements required for system operation
$\bar{n}$–# of failures during time t (average)
P(k,t)–Prob. of k failures during time t
i–element, s-subsystem subscript

General formulas:

$$A = \frac{MTBF}{MTBF + MTTR} \qquad \bar{n} = t/MTBF$$

$$U = 1 - A \qquad P(k,t) = \frac{e^{-\bar{n}} (\bar{n})^k}{k!}$$

Series Subsystems:

$$A_s = \prod_{1}^{N} A_i \qquad MTBF_s = 1/\sum_{1}^{N} (1/MTBF_i) \qquad MTTR_s = \frac{1 - A_s}{A_s} MTBF_s$$

Parallel Subsystems:

$$A_s = 1 - \prod_{1}^{N} (1 - A_i) \qquad MTBF_s = \frac{A_s}{1 - A_s} MTTR_s$$

$$MTTR_s = 1/\sum_{1}^{N} (1/MTTR_i)$$

Parallel "K-of-N"

$$MTTR_s = MTTR_i/(N-K+1)$$

$$MTBF_s = MTBF_i \left(\frac{MTBF_i}{MTTR_i}\right)^{(N-K)} \times \frac{(N-K)!\ (K-1)!}{N!}$$

INSTRUCTIONS

Press the RUN key to start the program, then follow the prompt by entering a number between 0 and 6.

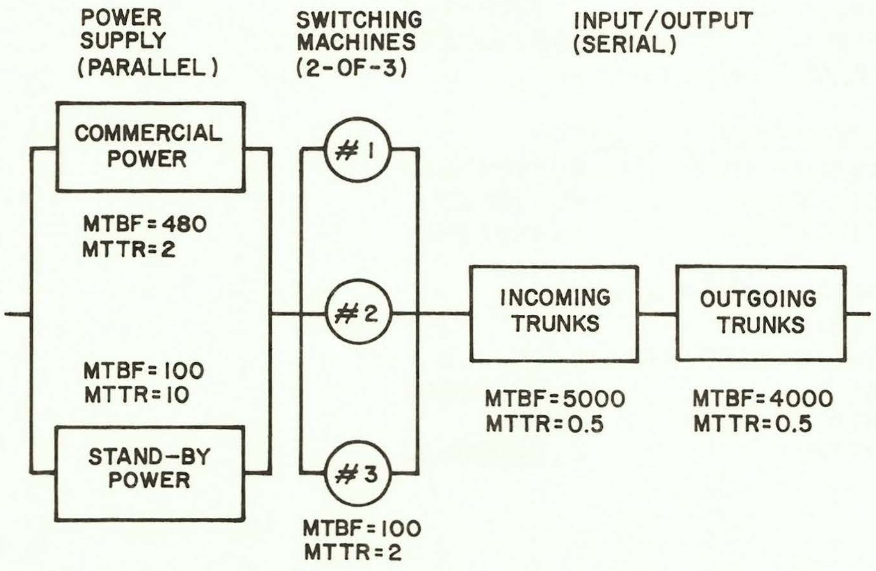
0 — Start a new run
1 — Enter MTBF and MTTR for a series subsystem
2 — Enter MTBF and MTTR for a redundant parallel subsystem
3 — Enter MTBF and MTTR for a redundant parallel k-of-n subsystem
4 — Clear subsystem if a mistake was made during entry
5 — Include a subsystem in an overall system
6 — Find probability of x failures during time t

EXAMPLE

Problem Definition

Consider a telephone office installation shown in Figure 21-1 consisting of a commercial and a stand-by power supply, three switching machines, two of which are required for operation, a group of incoming and a group of outgoing trunks connected to microwave antennas, all with known MTBFs and MTTRs. Find overall reliability parameters including probability of failure in 12 and 24 hours.

Figure 21-1



PROGRAM LISTING

```
OPERATION 0 - 6?2
PARALLEL
ENTER MTBF,MTTR?480,2
A=                    0.99585062
MTBF=                 480
MTTR=                 2

OPERATION 0 - 6?2
PARALLEL
ENTER MTBF,MTTR?100,10
A=                    0.99962278
MTBF=                 4416.6667
MTTR=                 1.6666667

OPERATION 0 - 6?5
ASYS=                 0.99962278
MTBFSYS=              4416.6667
MTTRSYS=              1.6666677

OPERATION 0 - 6?3
K-OF-N
ENTER MTBF,MTTR?100,2
ENTER K,N?2,3
A=                    0.99880144
MTBF=                 833.33333
MTTR=                 1

OPERATION 0 - 6?5
ASYS=                 0.99842467
MTBFSYS=              701.0582
MTTRSYS=              1.1061378

OPERATION 0 - 6?1
SERIES
ENTER MTBF,MTTR?5000,0.5
A=                    0.99990001
MTBF=                 5000
MTTR=                 0.50000025
```

```
OPERATION 0 - 6?1
SERIES
ENTER MTBF,MTTR?4000,0.5
A=                   0.99977504
A=                   0.99977504
MTBF=                2222.2222
MTTR=                0.50002885

OPERATION 0 - 6?5
ASYS=                0.99820007
MTBFSYS=             532.93112
MTTRSYS=             0.96097057

OPERATION 0 - 6?6
TIME,NO.FAILURES?24,1
P( 24,1 )=.043050892

OPERATION 0 - 6?6
TIME,NO.FAILURES?12,1
P( 12,1 )=.022015632


OPERATION 0 - 6?6
TIME,NO.FAILURES?24,2
P( 24,2 )=.00096937613
```

Discussion of Results

Looking at the subsystem first we see that the parallel power supply subsystem has a MTBF of 4416 hours, the 2-out-of-3 switching machine subsystem has a MTBF of 833 hours and the series incoming/outgoing trunk subsystem has a MTBF of 2222 hours. The overall system has a MTBF of 532 hours. To improve this factor would evidently require an improvement in reliability of the switching machines. The probability of one failure in 24 hours is 4.3%, of one failure in 12 hours is 2.2%. The probability of 2 failures in 24 hours is only 0.097%.

PROGRAMMING REMARKS

The program fills nearly the whole 2K memory of the Timex/Sinclair 1000. For this reason the prompts had to be kept to a minimum.

PROGRAM LISTING

```
 15 GOSUB 800
 20 LET M=1
 40 LET N=0
 50 PRINT
 60 PRINT "OPERATION 0 - 6?";
 70 INPUT Z
 80 PRINT Z
 90 GOTO 100*Z+10
110 PRINT "SERIES"
120 GOSUB 1000
130 LET C=C+1/A
140 LET D=D*A/(A+B)
150 LET F=1/C
160 LET H=F*(1-D)/D
170 LET E=D
180 PRINT "A=",E,"MTBF=",F,"MTTR=",H
190 GOTO 50
210 PRINT "PARALLEL"
215 GOSUB 1000
220 LET C=C+1/B
225 LET D=D*B/(A+B)
230 LET E=1-D
240 LET H=1/C
250 LET F=E*H/D
260 GOTO 180
310 PRINT "K-OF-N"
315 GOSUB 1000
320 PRINT "ENTER K,N?";
325 INPUT K
330 PRINT K;
335 INPUT R
340 PRINT ",";R
345 LET L=R-K
350 GOSUB 900
355 LET F=J
360 LET L=K-1
365 GOSUB 900
370 LET F=F*J
375 LET L=R
380 GOSUB 900
385 LET F=A*F/J*(A/B)**(R-K)
```

```
 390 LET H=B/(R-K+1)
 395 LET E=F/(F+H)
 400 GOTO 180
 410 GOSUB 800
 420 PRINT "SUBSYSTEM CLEARED"
 430 GOTO 50
 510 GOSUB 800
 520 LET M=M*E
 530 LET N=N+1/F
 540 LET P=(1-M)/M/N
 550 LET X=1/N
 560 PRINT "ASYS=",M,"MTBFSYS=",X,"MTTRSYS=",P
 570 GOTO 50
 610 PRINT "TIME,NO.FAILURES?";
 612 INPUT T
 614 PRINT T;
 615 INPUT Q
 620 PRINT ",";Q
 625 LET S=T*N
 630 LET L=Q
 635 GOSUB 900
 640 LET U=EXP -S*S**Q/J
 645 PRINT "P(";T;",";Q;")=";U
 650 GOTO 50
 800 LET C=0
 810 LET D=1
 820 RETURN
 900 LET J=1
 910 FOR I=1 TO L
 920 LET J=J*I
 930 NEXT I
 940 RETURN
1000 PRINT "ENTER MTBF,MTTR?";
1010 INPUT A
1020 PRINT A;
1030 INPUT B
1040 PRINT ",";B
1050 RETURN
```

MISCELLANEOUS

22. Frequency of Musical Notes on the Well-Tempered Scale

PROGRAM DESCRIPTION

This program finds the frequency of a given musical note on the well-tempered scale. The scale assumes a constant frequency ratio of the 12th root of 2 between two adjacent notes. Johann Sebastian Bach was so excited by the idea of well-tempered scale that he wrote a series of piano compositions in the key of each note in that scale. Octaves in the program are numbered as negative or positive integers, or 0, with note A in octave 0 having a frequency of 440-Hz. The frequency of notes doubles with each successively higher octave.

INSTRUCTIONS

Pressing the RUN key starts the program and issues two prompts for the note and for the octave number. The note should be entered as C, C+, D, D+, E, F, F+, G, G+, A, A+, or B; the octave as −2, −1, 0, 1, 2. . . etc. As the Timex/Sinclair 1000 does not have a sharp (#) key, a (+) is substituted for a (#).

EXAMPLE

Problem Definition

Find frequencies of A in octave 0, C# in the third octave, D in the −4 octave, and C and B in the first octave.

Sample Run

```
ENTER NOTE E.G. C,C+,D..?A
ENTER OCTAVE?0
A,OCTAVE 0=440 HZ

ENTER NOTE E.G. C,C+,D..?C+
ENTER OCTAVE?3
C+,OCTAVE 3=2217.4611 HZ

ENTER NOTE E.G. C,C+,D..?D
ENTER OCTAVE?-4
D,OCTAVE -4=18.354048 HZ

ENTER NOTE E.G. C,C+,D..?K
ENTER OCTAVE?3
WRONG NOTE

ENTER NOTE E.G. C,C+,D..?C
ENTER OCTAVE?1
C,OCTAVE 1=523.25113 HZ

ENTER NOTE E.G. C,C+,D..?B
ENTER OCTAVE?1
B,OCTAVE 1=987.7666 HZ
```

Discussion of Results

The first example, as expected, yields 440-Hz, the fourth entry shows a wrong input (K) which yields an error return and repeat of the prompt.

PROGRAMMING REMARKS

Here, for a change, is a short program. Lines 100 and 135 perform the error check of inputs. If the note designation is not recognized as legitimate, then the variable I retains its assigned value of 100 and the "WRONG NOTE" remark is displayed. The frequency computation is performed in line 170.

PROGRAM LISTING

```
 10 CLS
 50 PRINT
 52 PRINT "ENTER NOTE E.G. C,C+,D..?";
 60 INPUT B$
 70 PRINT B$
 80 PRINT "ENTER OCTAVE?";
 90 INPUT B
100 IF B<>INT B THEN GOTO 80
110 PRINT B
120 LET I=100
121 IF B$="C" THEN LET I=1
122 IF B$="C+" THEN LET I=2
123 IF B$="D" THEN LET I=3
124 IF B$="D+" THEN LET I=4
125 IF B$="E" THEN LET I=5
126 IF B$="F" THEN LET I=6
127 IF B$="F+" THEN LET I=7
128 IF B$="G" THEN LET I=8
129 IF B$="G+" THEN LET I=9
130 IF B$="A" THEN LET I=10
131 IF B$="A+" THEN LET I=11
132 IF B$="B" THEN LET I=12
135 IF I<>100 THEN GOTO 170
150 PRINT "WRONG NOTE"
160 GOTO 50
170 LET F=2**((I-10)/12+B)*440
180 PRINT B$;",OCTAVE ";B;"=";F;" HZ"
190 GOTO 50
```

23. Artificial Intelligence Strategy Game

PROGRAM DESCRIPTION

This program, which is a take-off on the popular penny-matching HEADS/TAILS game, is quite unique in that it learns by playing against the human opponent. It thus demonstrates a strong intelligence trait.

The game consists of the player calling either HEAD or TAIL. If the computer guesses correctly by making the same choice, then the computer wins. If the choice is different, then the player wins. If you, the player, play at random, then the computer also guesses at random and the number of wins and losses will be about the same. However, if you follow any of a number of simple strategies; e.g., play HEADS all the time, or play TAILS until you lose and then change to HEADS and vice versa, or play the same (HEADS or TAILS) if you lose, but alternate if you win, then the computer will outguess you after a few moves and beat you every time from then on, until you change your strategy. As soon as you change your strategy and the computer loses, it starts playing at random until it can lock on to your new strategy and win again.

The program checks two previous moves in the game and consults its memory for the same type of situation played earlier. If a former situation is found, then a guess is made by the computer based on previous experience. If the player then plays as expected, the computer wins. If the situation did not occur earlier, the computer plays randomly and stores the outcome of its move with the player's response to establish a new situation.

INSTRUCTIONS

Pressing the RUN key issues a prompt for a starter which initializes the RND BASIC function. The starter should be between 0 and 65535. Entering 0 will always start a different random sequence, otherwise the sequence is predetermined. The program then prompts for H(eads) or T(ails). It finally announces who won, shows the total score and continues with prompting for the next move.

EXAMPLE

Problem Definition

The player (you) will first follow the strategy of alternating when losing, playing the same when winning. Then, when the computer catches up with your strategy, you will reverse your strategy.

Sample Run

```
STARTER?1
(H)EAD OR (T)AIL?H
GAME 1,YOU WON, WIN COUNT=1
(H)EAD OR (T)AIL?H
GAME 2,YOU WON, WIN COUNT=2
(H)EAD OR (T)AIL?H
GAME 3,YOU LOST, WIN COUNT=1
(H)EAD OR (T)AIL?T
GAME 4,YOU WON, WIN COUNT=2
(H)EAD OR (T)AIL?T
GAME 5,YOU WON, WIN COUNT=3
(H)EAD OR (T)AIL?T
GAME 6,YOU LOST, WIN COUNT=2
(H)EAD OR (T)AIL?H
GAME 7,YOU LOST, WIN COUNT=1
(H)EAD OR (T)AIL?T
GAME 8,YOU LOST, WIN COUNT=0
(H)EAD OR (T)AIL?H
GAME 9,YOU LOST, WIN COUNT=-1
(H)EAD OR (T)AIL?T
GAME 10,YOU LOST, WIN COUNT=-2
(H)EAD OR (T)AIL?T
GAME 11,YOU WON, WIN COUNT=-1
(H)EAD OR (T)AIL?H
GAME 12,YOU WON, WIN COUNT=0
(H)EAD OR (T)AIL?T
GAME 13,YOU LOST, WIN COUNT=-1
(H)EAD OR (T)AIL?T
GAME 14,YOU LOST, WIN COUNT=-2
(H)EAD OR (T)AIL?T
GAME 15,YOU LOST, WIN COUNT=-3
(H)EAD OR (T)AIL?T
GAME 16,YOU LOST, WIN COUNT=-4
(H)EAD OR (T)AIL?T
GAME 17,YOU LOST, WIN COUNT=-5
(H)EAD OR (T)AIL?H
GAME 18,YOU WON, WIN COUNT=-4
(H)EAD OR (T)AIL?H
GAME 19,YOU LOST, WIN COUNT=-5
(H)EAD OR (T)AIL?T
GAME 20,YOU LOST, WIN COUNT=-6
```

```
(H)EAD OR (T)AIL?H
GAME 21,YOU WON, WIN COUNT=-5
(H)EAD OR (T)AIL?H
GAME 22,YOU LOST, WIN COUNT=-6
(H)EAD OR (T)AIL?T
GAME 23,YOU LOST, WIN COUNT=-7
(H)EAD OR (T)AIL?H
GAME 24,YOU LOST, WIN COUNT=-8
(H)EAD OR (T)AIL?
```

Discussion of Results

You follow your first strategy through game 10. The computer is winning each game, starting with game six. When you change strategy at game 11 you are first winning but the computer catches up again starting with game 13. You then revert to your old strategy in game 18. After a few moves the computer catches up with you again. After 24 games you are down 8 points.

PROGRAMMING REMARKS

The outcome of various plays are stored in the dimensioned variable A. The check for correct input (H) or (T) is performed in line 200. The program requires the SLOW mode so that results can be continuously displayed during the INKEY$ sequence.

PROGRAM LISTING

```
 10 CLS
 15 DIM A(8)
 18 SLOW
 20 PRINT "STARTER?";
 30 INPUT X
 40 PRINT X
 45 RAND (X)
 50 LET J=0
 60 LET I=0
 70 LET I=I+1
 80 IF I<3 THEN GOTO 140
 90 LET L=INT ((M*N+1)/2)
100 LET W=4*Q+2*L+P+1
110 IF A(W)=0 THEN GOTO 140
120 LET S=M*A(W)
```

```
130 GOTO 170
140 LET S=2*INT (2*RND)-1
170 PRINT "(H)EAD OR (T)AIL?";
180 IF INKEY$="" THEN GOTO 180
185 LET T$=INKEY$
190 IF INKEY$=T$ THEN GOTO 190
200 IF T$<>"T" AND T$<>"H" THEN GOTO 180
205 PRINT T$
210 LET U=1
220 IF T$="T" THEN LET U=-1
230 LET J=J-S*U
240 LET Z=2-I
250 IF Z=0 THEN GOTO 320
260 IF Z>0 THEN GOTO 340
270 IF A(W)=0 THEN GOTO 310
280 IF S=U THEN GOTO 320
290 LET A(W)=0
300 GOTO 320
310 LET A(W)=M*U
320 LET N=M
330 LET Q=P
340 LET M=U
350 LET P=INT ((S*U+1)/2)
360 LET A$="WON,"
370 IF S=U THEN LET A$="LOST,"
380 PRINT "GAME ";I;",YOU ";A$;" WIN COUNT=";J
390 GOTO 70
```

24. Multiple Dice Throw

PROGRAM DESCRIPTION

Many games; e.g., the popular YAHTZEE, and even computer modeling simulations, are based on the outcome of a multiple dice throw. This program permits a throw of several dice at once and displays the result of each multiple throw. The Timex/Sinclair 1000 BASIC pseudo random generator RND in line 90, which does the "throwing," works well in this application; however, if you have your own favorite, feel free to replace that line.

INSTRUCTIONS

Pressing the RUN key displays a prompt HOW MANY DICE for this throw. After a number n (one or more) is entered, the program displays n digits, where each digit is between one and six and corresponds to the spots on a die. Then the next prompt appears on the display for another throw.

EXAMPLE

Problem Definition

Keep throwing dice, as shown in the listing.

Sample Run

```
MULTIPLE DICE

HOW MANY DICE ?5
4,2,2,3,5
HOW MANY DICE ?3
6,6,1
HOW MANY DICE ?1
1
HOW MANY DICE ?1.5
HOW MANY DICE ?0
HOW MANY DICE ?10
3,4,3,6,3,2,1,6,4,3
```

```
HOW MANY DICE ?2
6,6
HOW MANY DICE ?3
2,5,4
HOW MANY DICE ?7
3,5,6,5,4,2,4
HOW MANY DICE ?5
6,3,6,2,5
HOW MANY DICE ?6
5,4,2,3,3,3
```

Discussion of Results

The results are displayed. When we requested 0 or 1.5 dice, since both numbers are not acceptable, the prompt reappears.

PROGRAMMING REMARKS

Error check of the input is performed in line 70. The program is straightforward and short. Line 90 indicates how to transform the random number generated by RND from the range 0-1 into the range of integers between 1 and 6.

PROGRAM LISTING

```
 10 CLS
 20 PRINT "MULTIPLE DICE"
 30 PRINT
 40 PRINT "HOW MANY DICE ?";
 50 INPUT A
 60 PRINT A
 70 IF A<>INT A OR A<1 THEN GOTO 40
 80 FOR I=1 TO A
 90 PRINT INT (6*RND)+1;
100 IF I=A THEN GOTO 120
110 PRINT ",";
120 NEXT I
130 GOTO 30
```

25. Two-User Checking Account

PROGRAM DESCRIPTION

Here is a program to aid in keeping the financial side of a household running smoothly. In these days of two working spouses the problem frequently arises how to handle a joint checking account. Both parties may keep separate checkbooks in which they record separately all their transactions such as deposits, checks being written against the account, and transfers between the two spouses. When the bank sends a statement at the end of the month the following program will help in balancing the two checkbooks. The equations used are as follows:

Spouse #2 balance = previous month's balance of spouse #2 +
deposits of spouse #2 − cleared checks of spouse #2 +
transfers from spouse #1 to #2, also

Spouse #2 balance = current bank balance −
checkbook balance #1 − non-cleared checks #1 −
earned interest (for NOW accounts) + bank charges.

The spouse #2 balance is computed by the program in two ways and the results are compared. If they are the same then the account is balanced. If they are different, then there is an error either in the bank statement or, more probably, in one of the checkbook entries.

INSTRUCTIONS

Pressing the RUN key issues a prompt for one of the following commands:

Z — Clears all registers, starts program
X — Prompts for previous balance of spouse #2
C — Prompts for each deposit to joint account by spouse #2
V — Prompts for each cleared check by spouse #2
B — Prompts for each internal transfer from account of spouse #1 to #2; use negative values for transfers from #2 to #1.
N — Prompts for current checkbook balance of spouse #1
M — Prompts for non-cleared checks written by #1
A — Prompts for the current bank balance
S — Prompts for earned interest (NOW checking accounts)
K — Prompts for bank charges, if any
D — Displays spouse #2 balance computed both ways

EXAMPLE

Problem Definition

The monthly bank statement just arrived showing a balance in the joint account of $1234.45. Spouse #1 has a checkbook balance of $751.23, and two checks written by #1 for $15.25 and $10.00 did not yet clear. The bank credited the account with $6.75 in interest. Spouse #2 during the month wrote three checks for $250.00, $125.00, and $50.00, but the last check did not yet clear. Spouse #2 had deposits credited for $150.00 and $175.00, and an internal transfer from spouse #1 for $300.00. The previous checkbook balance of spouse #2 was $203.22 and bank charges came to exactly $2.00. Find the current checkbook balance of spouse #2 and see that the account is balanced.

Sample Run

```
2 PERSON CHECKING

ENTER Z,X,C,V,B,N,M,A,S,K OR D?X
PREV BAL NO. 2?203.22

ENTER Z,X,C,V,B,N,M,A,S,K OR D?C
DEPOSITS NO. 2?150,TOT=150

ENTER Z,X,C,V,B,N,M,A,S,K OR D?C
DEPOSITS NO. 2?175,TOT=325

ENTER Z,X,C,V,B,N,M,A,S,K OR D?V
CL. CHECKS NO. 2?250,TOT=250

ENTER Z,X,C,V,B,N,M,A,S,K OR D?V
CL. CHECKS NO. 2?125,TOT=375

ENTER Z,X,C,V,B,N,M,A,S,K OR D?B
XFERS 1 TO 2?300,TOT=300

ENTER Z,X,C,V,B,N,M,A,S,K OR D?N
BAL NO.1?751.23

ENTER Z,X,C,V,B,N,M,A,S,K OR D?M
OUTST CHKS NO.1?15.25,TOT=15.25

ENTER Z,X,C,V,B,N,M,A,S,K OR D?M
OUTST CHKS NO.1?10,TOT=25.25
```

```
ENTER Z,X,C,V,B,N,M,A,S,K OR D?A
BANK BAL?1234.45

ENTER Z,X,C,V,B,N,M,A,S,K OR D?S
INTEREST?6.75

ENTER Z,X,C,V,B,N,M,A,S,K OR D?K
BANK CHARGES?2,TOT=2

ENTER Z,X,C,V,B,N,M,A,S,K OR D?D
BAL NO. 2=453.22,453.22
ACCOUNT BALANCED
```

Discussion of Results

The account fortunately balanced; both computations led to the same result. Notice that the $50 check written by spouse #2, which did not clear yet, did not have to be entered.

PROGRAMMING REMARKS

Branching is performed via the INKEY$ function and input is then compared with the recognized commands Z, X, C... Line 580 takes care of the roundoff error of the Timex/Sinclair 1000. Notice that in the subroutine starting in line 1000 all variables are set to 0. Why not use the CLEAR command instead? The reason is that then statements such as those in line 190 or 250 would return error 2 (undefined variable).

PROGRAM LISTING

```
10 CLS
15 PRINT "2 PERSON CHECKING"
20 SLOW
30 GOSUB 1000
40 PRINT
45 PRINT "ENTER Z,X,C,V,B,N,M,A,S,K OR D?";
50 IF INKEY$="" THEN GOTO 50
55 PAUSE 5
60 LET A$=INKEY$
65 IF A$=INKEY$ THEN GOTO 65
70 PRINT A$
90 IF A$<>"Z" THEN GOTO 110
```

```
 95 GOSUB 1000
100 PRINT "SYSTEM CLEARED"
105 GOTO 40
110 IF A$<>"X" THEN GOTO 160
120 PRINT "PREV BAL NO. 2?";
130 INPUT X
140 PRINT X
150 GOTO 40
160 IF A$<>"C" THEN GOTO 220
170 PRINT "DEPOSITS NO. 2?";
180 INPUT P
190 LET C=C+P
200 PRINT P;",TOT=";C
210 GOTO 40
220 IF A$<>"V" THEN GOTO 280
230 PRINT "CL. CHECKS NO. 2?";
240 INPUT Y
250 LET V=V+Y
260 PRINT Y;",TOT=";V
270 GOTO 40
280 IF A$<>"B" THEN GOTO 340
290 PRINT "XFERS 1 TO 2?";
300 INPUT L
310 LET B=B+L
320 PRINT L;",TOT=";B
330 GOTO 40
340 IF A$<>"N" THEN GOTO 390
350 PRINT "BAL NO.1?";
360 INPUT N
370 PRINT N
380 GOTO 40
390 IF A$<>"M" THEN GOTO 450
400 PRINT "OUTST CHKS NO.1?";
410 INPUT J
420 LET M=M+J
430 PRINT J;",TOT=";M
440 GOTO 40
450 IF A$<>"A" THEN GOTO 490
455 PRINT "BANK BAL?";
460 INPUT A
470 PRINT A
480 GOTO 40
490 IF A$<>"S" THEN GOTO 540
500 PRINT "INTEREST?";
```

```
 510 INPUT S
 520 PRINT S
 530 GOTO 40
 540 IF A$<>"D" THEN GOTO 610
 550 LET F=X+C-V+B
 560 LET G=A-N-M-S+K
 570 PRINT "BAL NO. 2=";F;",";G
 580 IF ABS (F-G)<.001 THEN PRINT
     "ACCOUNT BALANCED"
 590 GOTO 40
 610 IF A$<>"K" THEN GOTO 660
 620 PRINT "BANK CHARGES?";
 630 INPUT F
 640 LET K=K+F
 650 PRINT F;",TOT=";K
 655 GOTO 40
 660 PRINT "REPEAT"
 670 GOTO 40
1000 LET C=0
1010 LET V=0
1020 LET B=0
1030 LET M=0
1040 LET K=0
1050 LET X=0
1060 LET A=0
1070 LET N=0
1080 LET S=0
1090 RETURN
```

APPENDIX

Programming Hints and Techniques

While writing and debugging programs in this book I came up with a number of programming hints and techniques that I would like to share with other Timex/ Sinclair 1000 users to save them some grief and frustration. They are listed here.

1. Roundoff
If you deal with floating point numbers do not make a decision on equalities of two numbers (e.g., IF X = Y THEN . . .). The TS 1000 handles roundoff strangely; e.g.,

```
100 LET X=3.25-3-.25
110 PRINT X
```

will display 1.164 . . . E − 10. Therefore always take the absolute value of the difference and compare it with a small number.

2. Running Out of Memory
When you write a long program which causes the Timex/Sinclair 1000 to run out of memory, the computer will start dying "gracefully" and will make your life miserable. One of the first symptoms of memory "tightness" is when the computer ignores the EDIT function (Shift 1). However, as the Timex/Sinclair 1000 shares display and program memory you can get around this problem. For example, if you want to edit line 350 while you are running out of memory, do the following:

LIST 350	Listing appears on the screen with line 350 on top
CLS	Screen goes blank
EDIT	Presto, Line 350 appears on the bottom of the screen ready to be edited

3. Multiple Branches
The 1000 lacks a computed GOTO. To get around this deficiency, number various branches of your program in multiples of 100 and use the statement

```
GOTO 100*A
```

The variable A should of course be an integer.

4. Leaving a Loop

If you are stuck during program execution in a tight loop with an INPUT statement in it, you may not be able to get out of the loop without unplugging the computer. The BREAK key will simply be interpreted by INPUT as a space. If the expected input is numerical you can still get out of the loop by keying in, for example, Shift Enter (Function) LN 0, and getting an "A" error return. If the expected input is alphanumeric, then your only hope is to try to fill the screen and to get out with a "5" error return.

5. INPUT Statement

Precede all INPUTs with a prompt. An INPUT statement does not produce a question mark automatically as in some other BASICs. Follow an INPUT statement with a PRINT to display what you have just entered. Otherwise you will never know. A typical sequence could read as follows:

```
10 PRINT "ENTER X?";
20 INPUT X
30 PRINT X
```

6. Use of INKEY$

INKEY$ is used as a convenient means for program branching. Within a loop with INKEY$ the computer should be in the SLOW mode or the program should have PAUSE statements, otherwise the screen will remain blank with no visible prompts. Here is a foolproof sequence to detect, for example, the "X" key. It will operate in both SLOW and FAST mode:

```
100 PRINT "WHAT NEXT?";
110 PAUSE 10
120 IF INKEY$ = " " THEN GOTO 110
130 LET A$ = INKEY$
140 PRINT A$
150 PAUSE 10
160 IF A$ = INKEY$ THEN GOTO 150
170 IF A$ < >"X" THEN GO TO 100
180 REM "X" HAS BEEN DETECTED
```

7. Use of Quotes

To indicate an alphanumeric null use Shift P (") twice and not Shift Q (" "). Shift Q can only be used to display quotes inside alphanumeric strings.

8. Variable Initialization

Every variable has to be initialized with an assignment or a DIM statement before it can be put on the right side of an assignment statement.Otherwise the program execution will stop with a "2" error. Though annoying at first, this feature is great for program debugging. It will flag any misspelled or uninitialized variables. But beware of the CLEAR command which will obliterate all variable initializations (assignment or DIM). In fact I have found very little use for the CLEAR command.

9. Dimensioning of Strings

Be careful when using dimensioned strings. A singly dimensioned string can be only one character long. The second dimension is interpreted by the Timex/Sinclair BASIC as the length of the string.

10. Generating Similar Program Statements

A fast way to generate several similar statements in your program is to write the first statement, then edit it and also edit the program line number. Editing a program line number generates two statements — one with the old program line number, one with the new one. The Timex/Sinclair's BASIC permits editing of line numbers, while most other BASICs do not.

11. Deleting the Input Buffer

When entering data or programs from the keyboard the characters are temporarily stored in a buffer and are displayed on the bottom of the screen. Pressing the ENTER key transfers the buffer into main memory. If at some point you decide to erase the current buffer you may do it quickly by pressing Shift 1 (EDIT) rather than deleting it character by character or token by token with the Shift 0 (DELETE) key.

Spectrum Conversions

The ZX Spectrum is a further step up the ladder of the Sinclair's home computer family. It retains all the good features of the ZX-81 and the Timex/Sinclair 1000 and adds many new ones. The ZX Spectrum computer in its basic configuration comes with 16K of RAM instead of 2K in the Timex/Sinclair 1000, has color and extended graphic capabilities, and adds several new BASIC functions. Fortunately most ZX-81 and Timex/Sinclair 1000 BASIC functions are identical to those featured in the ZX Spectrum. Therefore, most of the programs featured in this book will run on the ZX Spectrum exactly as shown, without any modifications.

However, for better readability all variable names shown in this book in upper case should be entered in the ZX Spectrum in lower case, its standard operating mode. Certain functions have been slightly changed in the ZX Spectrum in function or in name. For example, the RAND function used in program #14 and #23 is called RANDOMIZE. The SLOW and FAST functions used in several programs in this book do not exist on the ZX Spectrum and should be omitted. Another minor change is replacement of the exponentiation sign (**) by an uparrow. One difference between the two computers which should be carefully watched is the change in values of the CODE function. The CODE function in the ZX Spectrum is very similar to the ASC function found on most other BASIC computers. For proper operation the following changes should be made in programs #6 and #8 which make use of the CODE function.

Program #6. Hex-to-Decimal and Decimal-to-Hex Conversion.

Enter hexadecimal digits A-F in upper case. Replace lines 260 and 510 and add lines 261, 262, and 511 as follows:

```
260 LET c = CODE a$(l-i)
261 IF c < = 57 THEN LET c = c - 48
262 IF c > 57 THEN LET c = c - 55
510 IF e < = 9 THEN LET b$(k) = CHR$(e + 48)
511 IF e > 9 THEN LET b$(k) = CHR$(e + 55)
```

Program #8. Relative Offset Computations for Machine Code Programming.

Enter hexadecimal digits A-F in upper case. Replace lines 280, 290, and 370 and add lines 281, 291, and 371 as follows:

```
280 IF g < = 9 THEN LET e$ = CHR$(g + 48)
281 IF g > 9 THEN LET e$ = CHR+(g + 55)
290 IF g < 9 THEN LET w$ = CHR$(z - 16 * g + 48)
291 IF g > 9 THEN LET w$ = CHR$(z - 16 * g + 55)
370 IF CODE c$(l-i) < = 57 THEN LET r = CODE c$(l -i) - 48
371 IF CODE c$(l-i) > 57 THEN LET r = CODE c$(l -i) - 55
```

References and Further Reading

1. *Science and Engineering Sourcebook,* Cass R. Lewart, Prentice-Hall/Micro Text. Collection of programs for the TRS-80 PC/Sharp PC-1211 pocket computers.

2. *Handbook of Mathematical Functions,* U.S. Department of Commerce, National Bureau of Standards, edited by Milton Abramowitz and Irene A. Stegun. Best collection of mathematical formulas.

3. *Introduction to Scientific Computing,* Gabriel A. Pall, Appleton-Century-Crofts. A good explanation of Runge-Kutta and other numerical approximations.

4. *Reference Data for Radio Engineers,* International Telephone and Telegraph Corporation, fourth and fifth editions. Excellent compendium of formulas relating to electrical engineering.

5. "Predicting Queue Performance on a Programmable Handheld Calculator," Ronald Zussman, *Computer Design,* August, 1978. A queuing parameter program for the TI-59.

6. *Fundamentals of Queuing Theory,* Donald Gross and Carl M. Harris, J. Wiley & Sons. An in-depth look and derivation of queuing equations.

7. "Forecast Computer System Reliability with a Handheld Programmable Calculator," Ronald Zussman, *Computer Design,* March, 1979. A reliability parameter program for the TI-59.

8. "Reliability Computations on a Handheld Programmable Calculator," Cass R. Lewart, *Computer Design,* November, 1979. A reliability parameter program for the HP-67.

9. "Calc Program Assembles Machine Code," Cass R. Lewart and Daniel S. Lewart, *Machine Design,* March 12, 1981. Relative offset computations program for the HP-67.

10. "Pocket Computer Solves for LC Resonance Using BASIC," Cass R. Lewart, *Electronics,* June 16, 1981, p. 189.

11. "Pocket Computer Tackles Queuing Problems in BASIC," Cass R. Lewart, *Electronics,* September 22, 1981, p. 157.

12. "Pocket Computer Scales Computer Generated Plots," Cass R. Lewart, *Electronics,* December 15, 1981, p. 159.

Index

General Interest
Computers

Twenty-five science and engineering programs for the Timex/Sinclair 1000 bring professional sophistication to the world of low-cost computing!

With detailed documentation and liberally illustrated by sample runs, these programs solve the most frequently encountered problems in the areas of:

- ★ electrical engineering
- ★ data transmission
- ★ number theory
- ★ computer programming
- ★ computer-generated plotting
- ★ probability and statistics
- ★ mathematics
- ★ operations research

These compact, clearly-written programs are compatible with the Timex/Sinclair Spectrum computer, and can easily be translated to run on other BASIC computers.

About the Author

Cass Lewart is well known in electronics publishing. He is the author of a book of programs, and over 100 articles in magazines such as 80-Microcomputing, Computer Design, Machine Design, Electronics, EDN, Radio-Electronics, Popular Electronics, Elementary Electronics, Popular Science, and Microwaves. Mr. Lewart is a graduate of the Federal Institute of Technology in Zurich, Switzerland, and is a research engineer in communications and computer science.

Micro Text Publications, Inc.
One Lincoln Plaza
New York, NY 10023
AND
McGraw-Hill Book Company
Serving the Need for Knowledge
1221 Avenue of the Americas
New York, NY 10020

ISBN 0-07-037444-9